FIG. 1

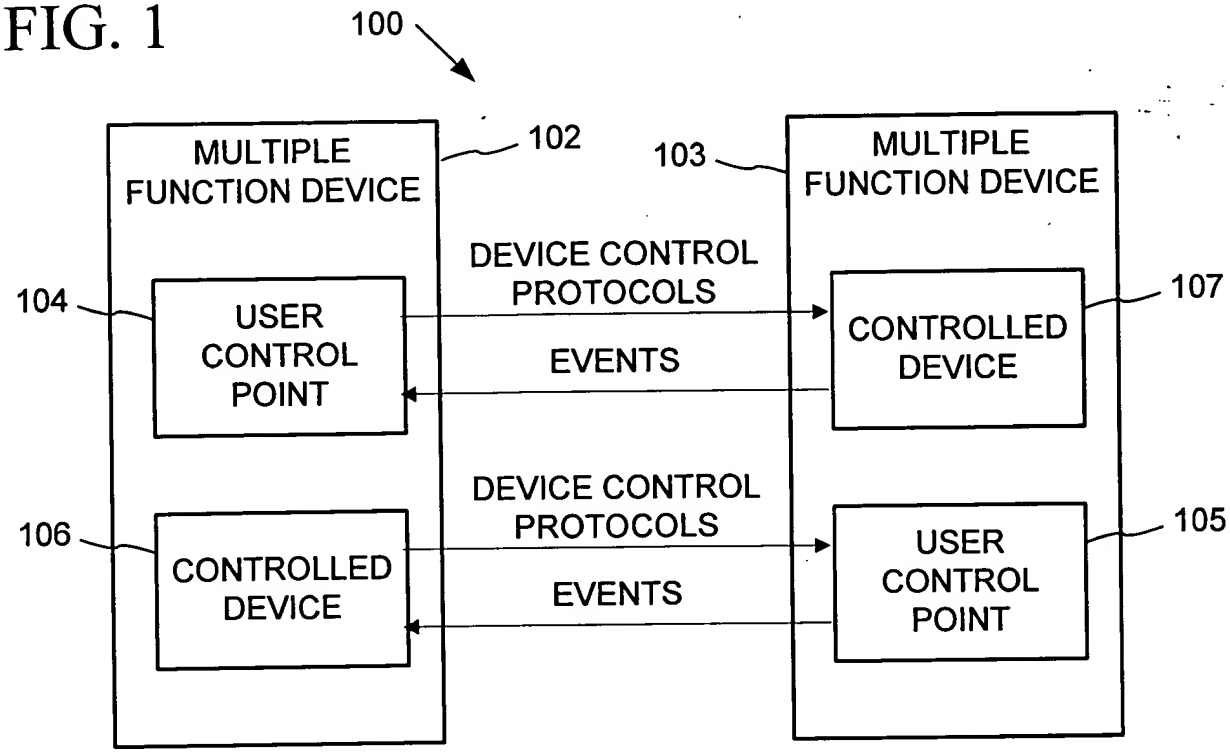

FIG. 2

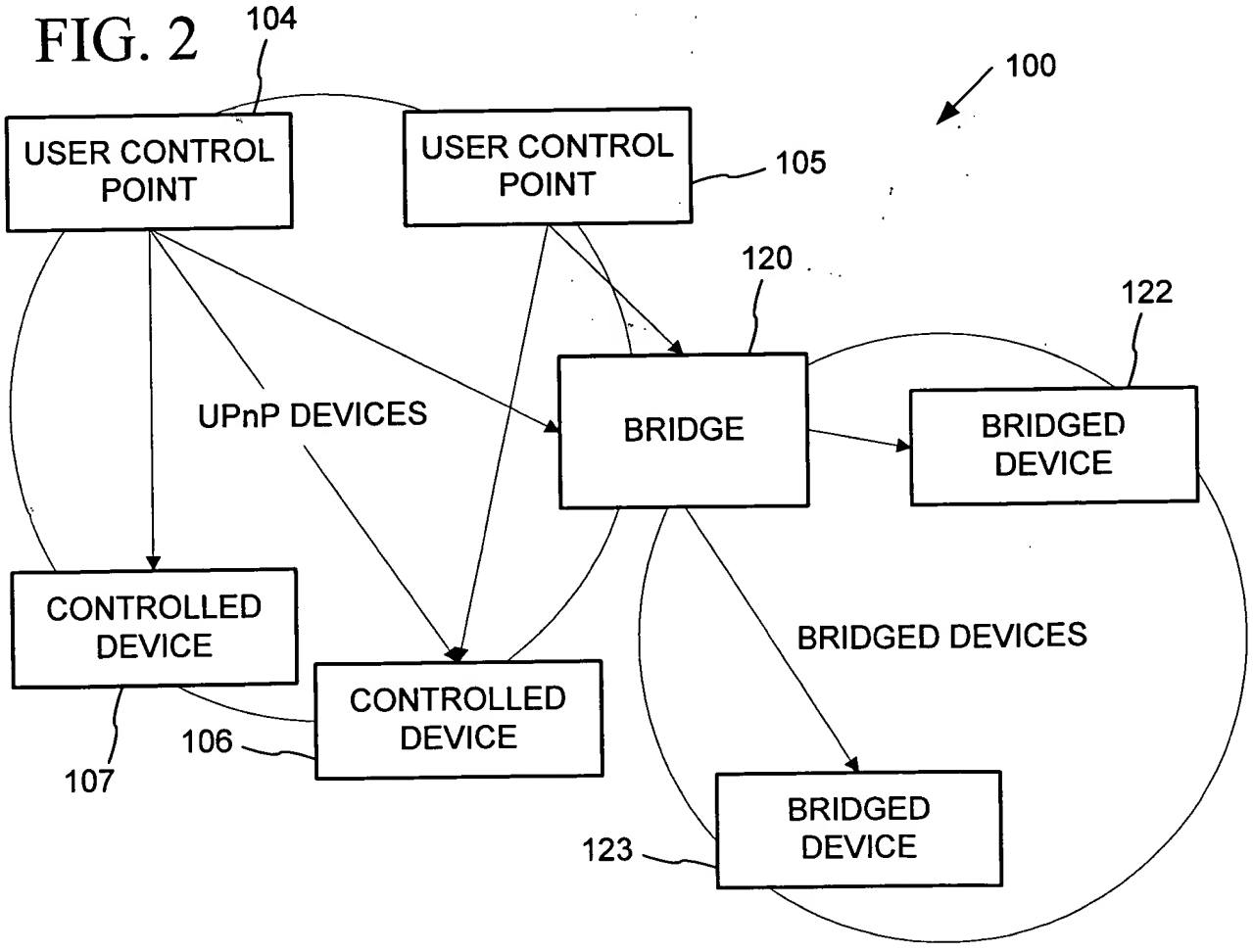

FIG. 3

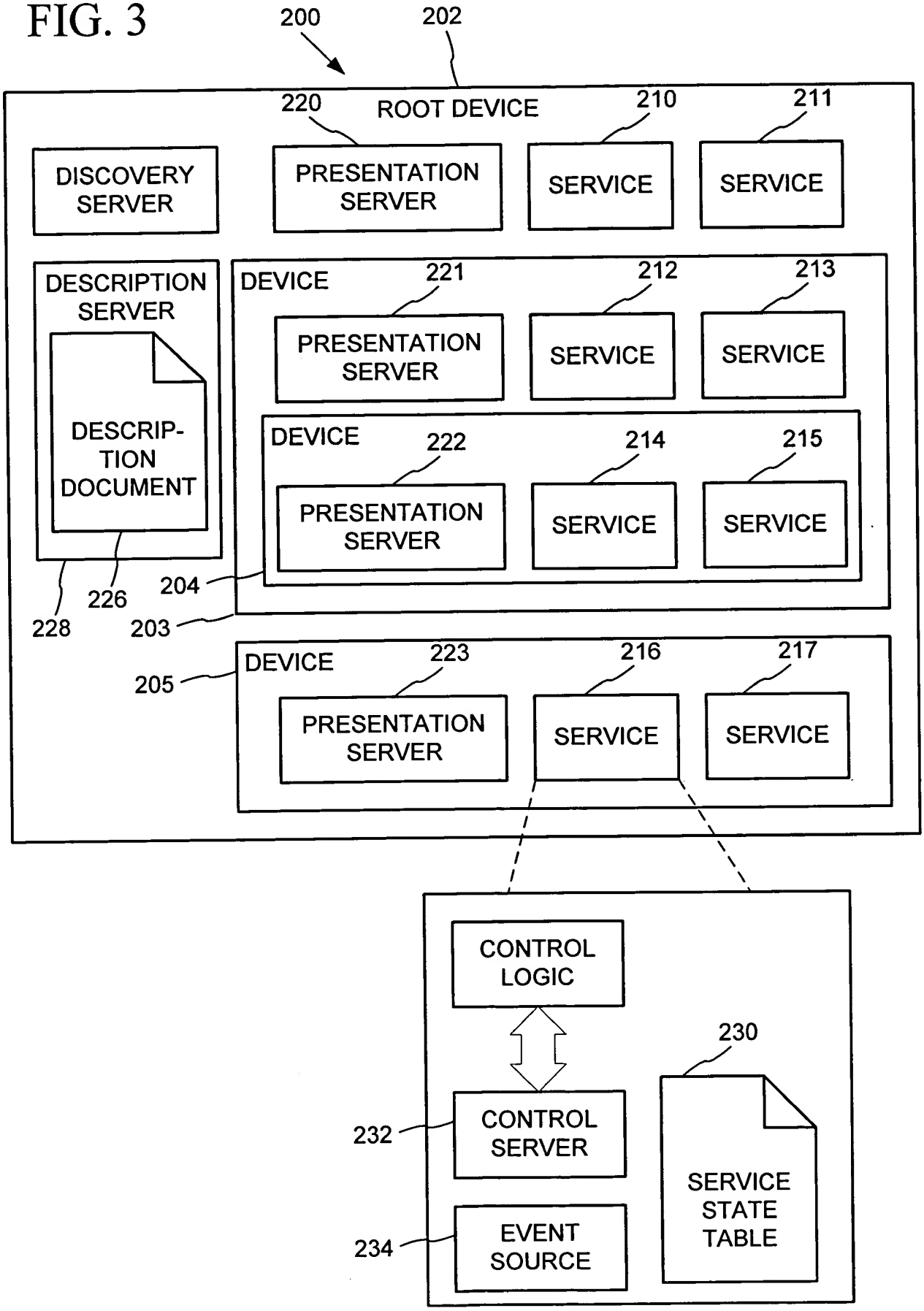
200
202
ROOT DEVICE
220
210
211
DISCOVERY SERVER
PRESENTATION SERVER
SERVICE
SERVICE
DESCRIPTION SERVER
DESCRIP-TION DOCUMENT
DEVICE
221
212
213
PRESENTATION SERVER
SERVICE
SERVICE
DEVICE
222
214
215
PRESENTATION SERVER
SERVICE
SERVICE
204
203
226
228
DEVICE
205
223
216
217
PRESENTATION SERVER
SERVICE
SERVICE
CONTROL LOGIC
230
232
CONTROL SERVER
SERVICE STATE TABLE
234
EVENT SOURCE

FIG. 4

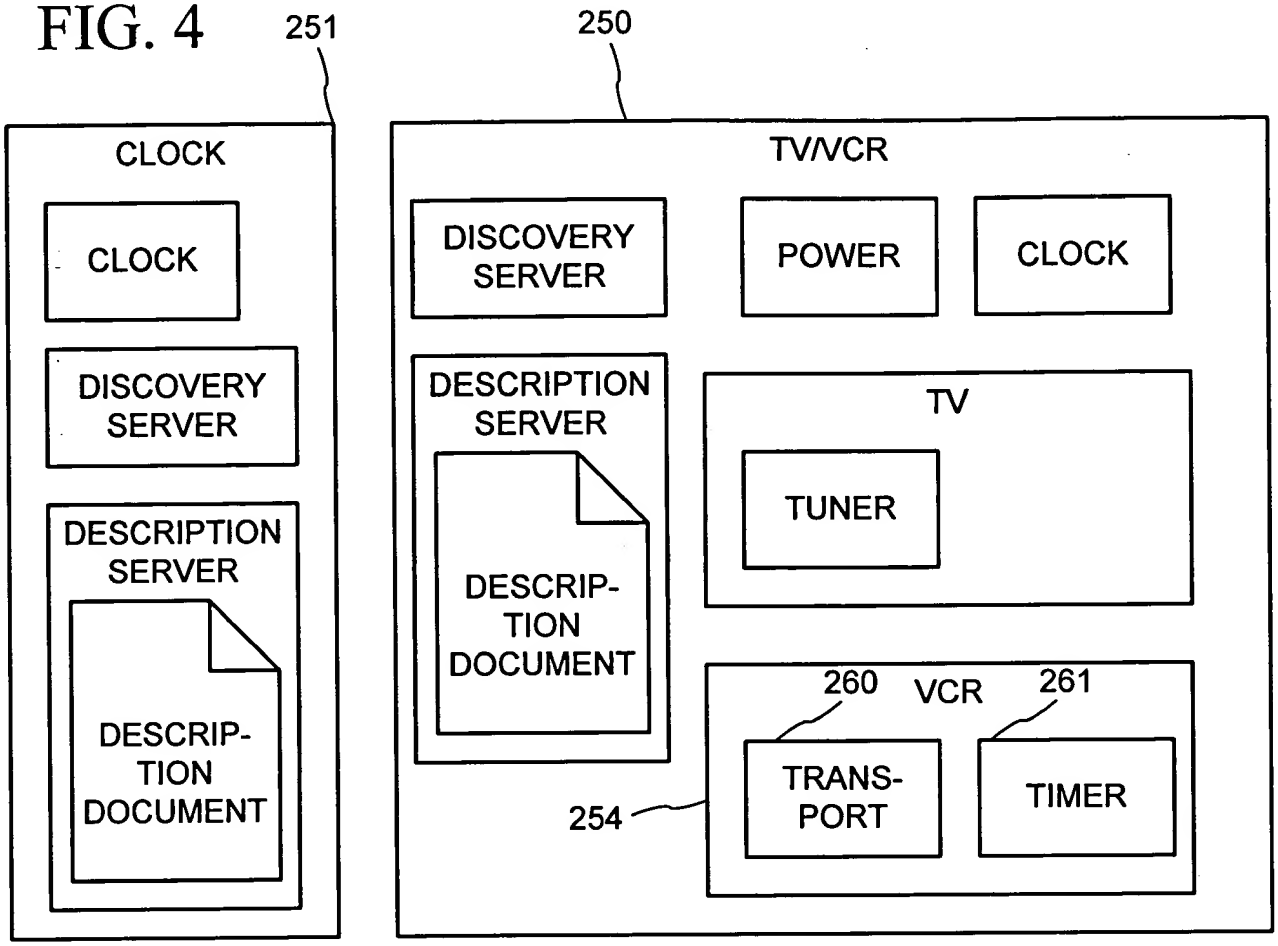
251
250
CLOCK
CLOCK
DISCOVERY SERVER
DESCRIPTION SERVER
DESCRIP-TION DOCUMENT
TV/VCR
DISCOVERY SERVER
POWER
CLOCK
DESCRIPTION SERVER
DESCRIP-TION DOCUMENT
TV
TUNER
260
VCR
261
TRANS-PORT
TIMER
254

FIG. 5

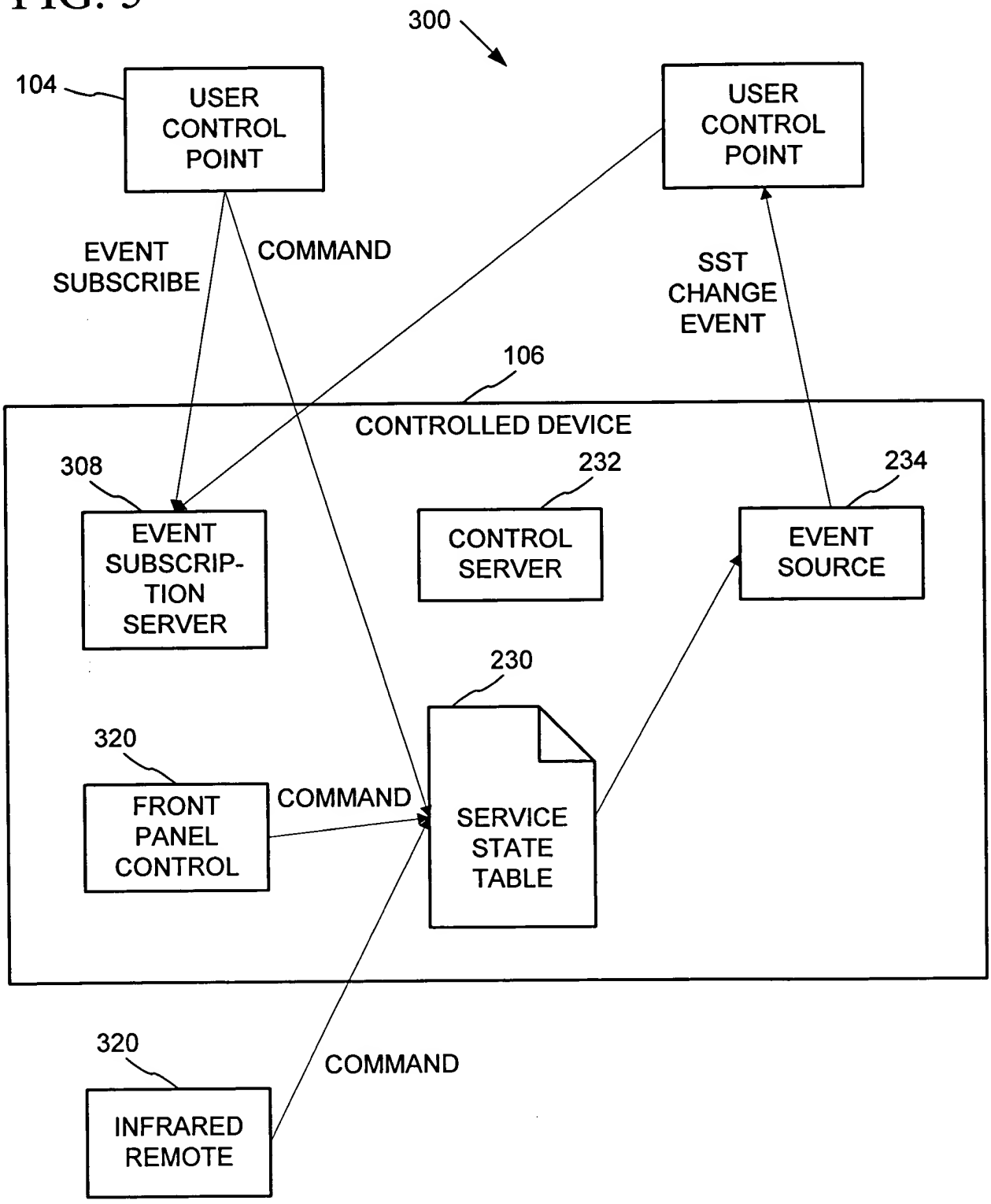
300
104
USER CONTROL POINT
USER CONTROL POINT
EVENT SUBSCRIBE
COMMAND
SST CHANGE EVENT
106
CONTROLLED DEVICE
308
EVENT SUBSCRIP-TION SERVER
232
CONTROL SERVER
234
EVENT SOURCE
230
320
FRONT PANEL CONTROL
COMMAND
SERVICE STATE TABLE
320
INFRARED REMOTE
COMMAND

FIG. 6

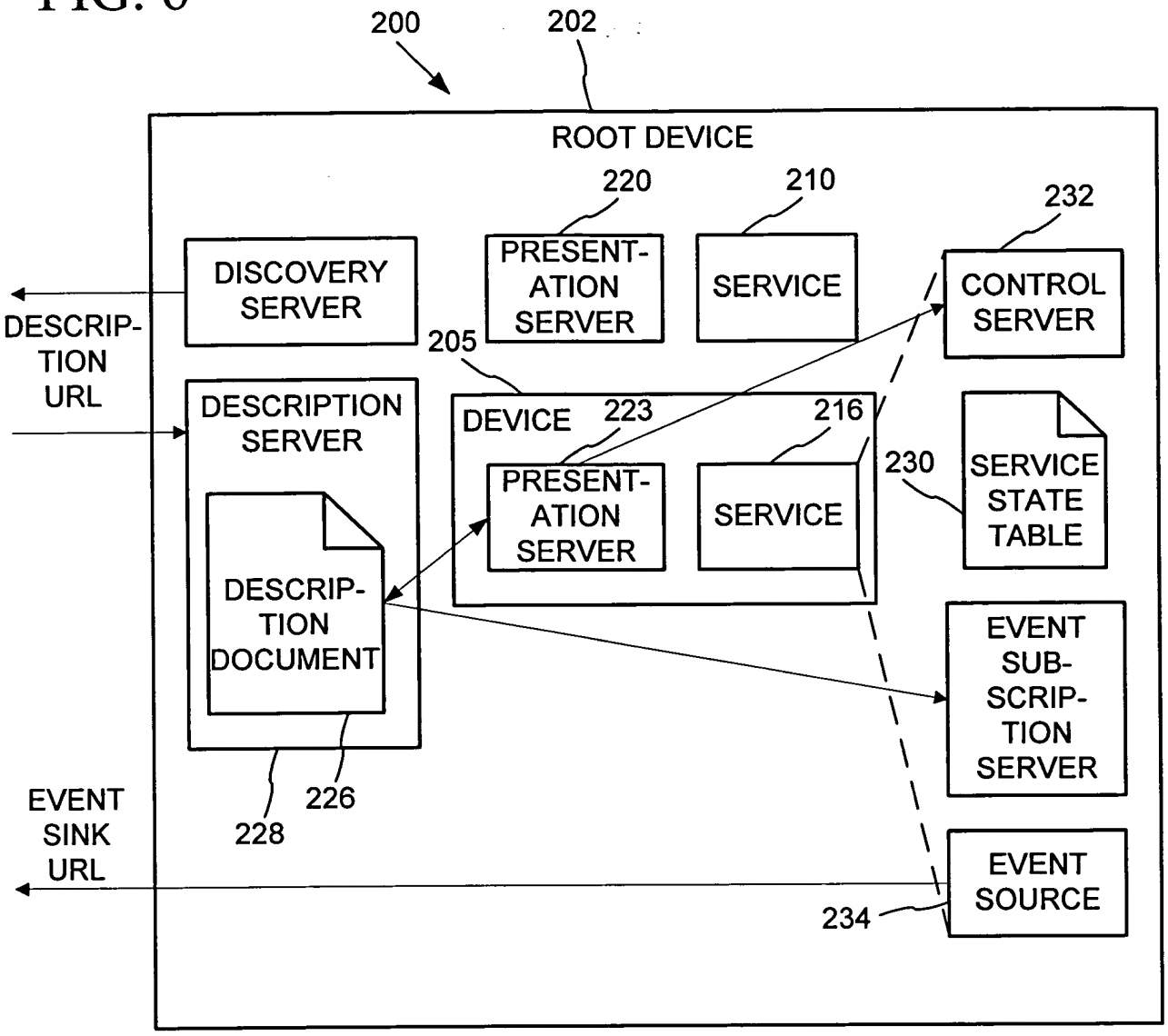
200
202
ROOT DEVICE
220
210
232
DISCOVERY SERVER
PRESENT-ATION SERVER
SERVICE
CONTROL SERVER
DESCRIP-TION URL
205
DESCRIPTION SERVER
DEVICE
223
216
230
SERVICE STATE TABLE
PRESENT-ATION SERVER
SERVICE
DESCRIP-TION DOCUMENT
EVENT SUB-SCRIP-TION SERVER
EVENT SINK URL
228
226
EVENT SOURCE
234

FIG. 7

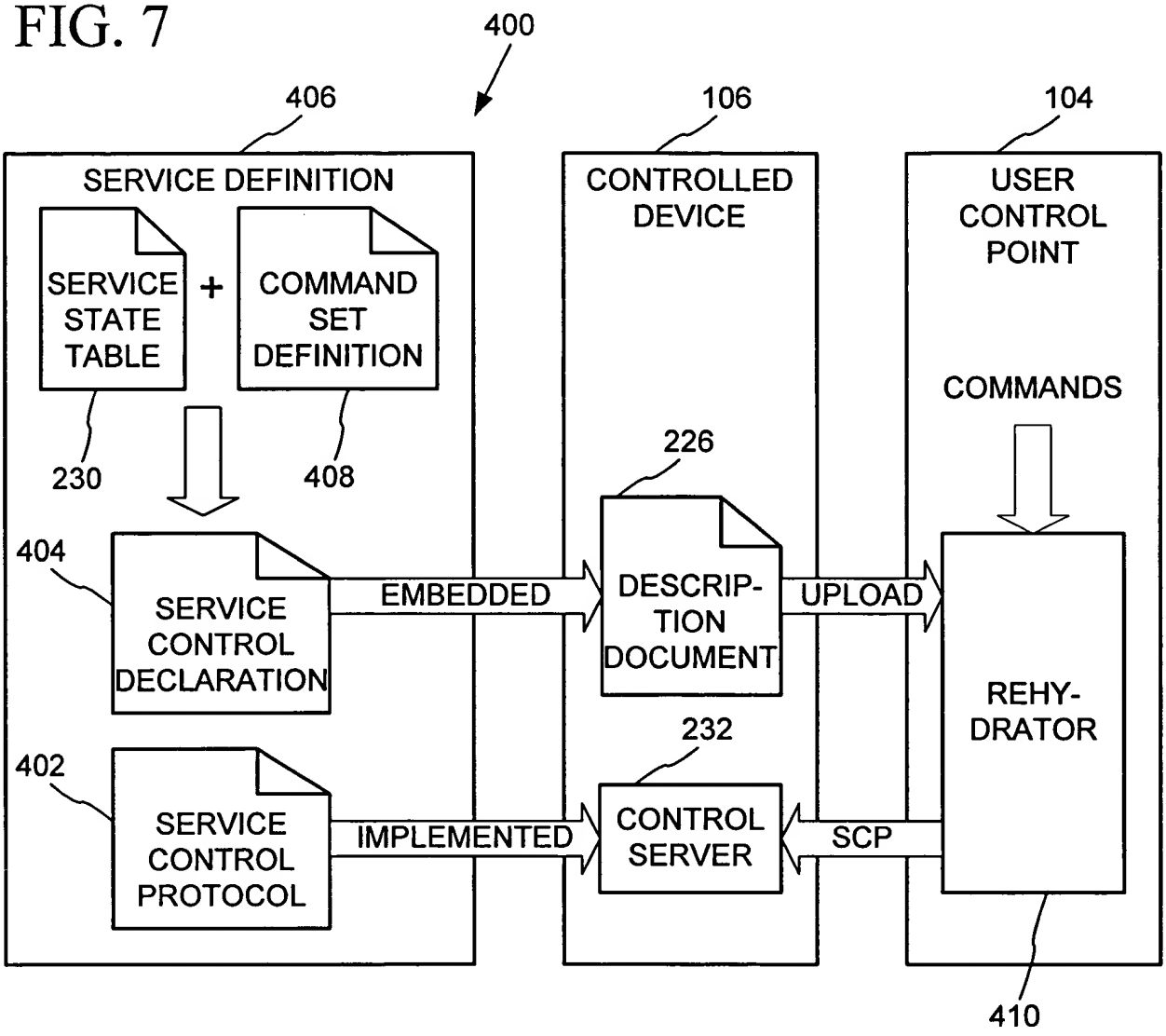
400
406
106
104
SERVICE DEFINITION
CONTROLLED DEVICE
USER CONTROL POINT
SERVICE STATE TABLE
+
COMMAND SET DEFINITION
230
408
COMMANDS
226
404
SERVICE CONTROL DECLARATION
EMBEDDED
DESCRIP-TION DOCUMENT
UPLOAD
REHY-DRATOR
232
402
SERVICE CONTROL PROTOCOL
IMPLEMENTED
CONTROL SERVER
SCP
410

FIG. 8

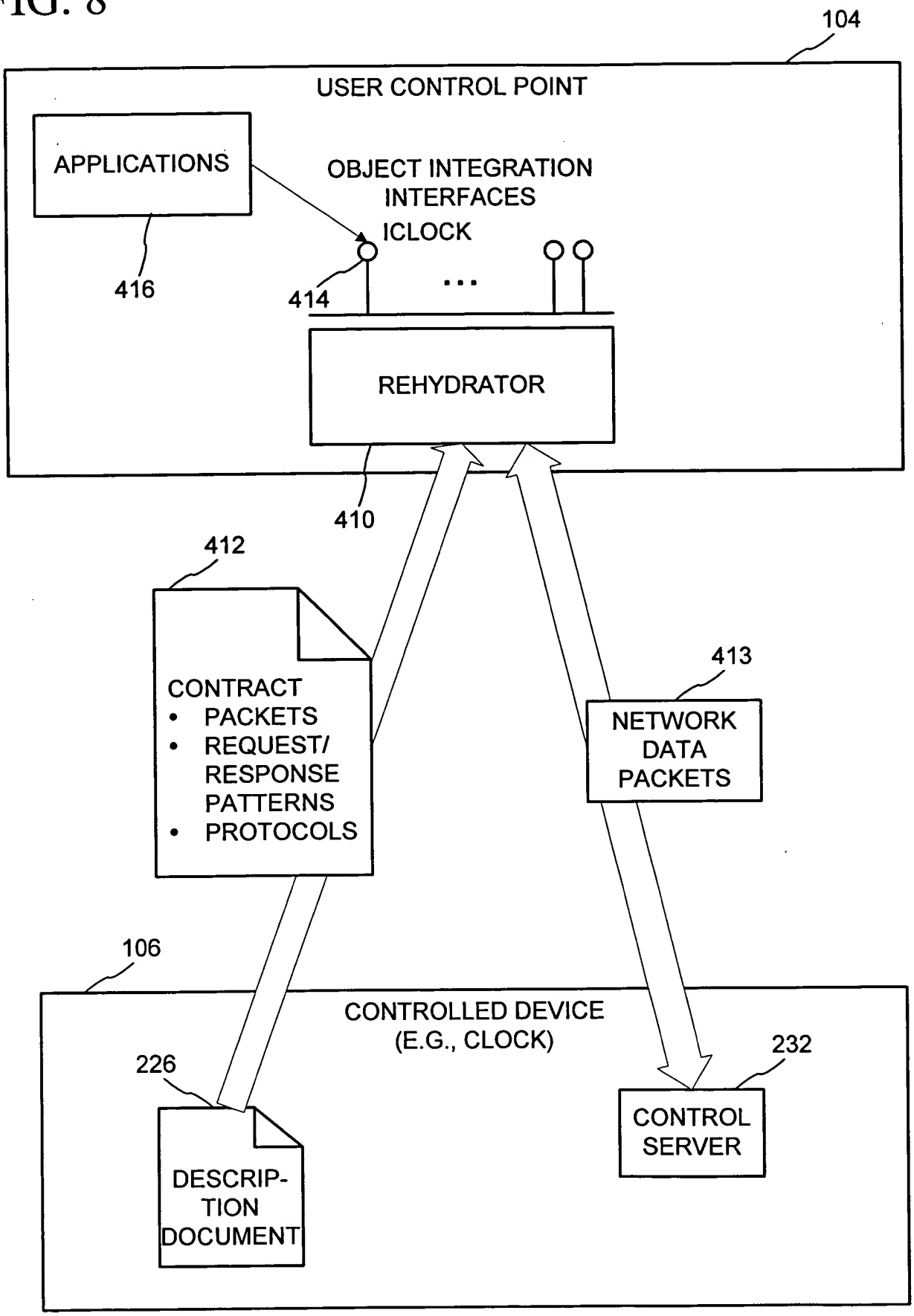
104
USER CONTROL POINT
APPLICATIONS
OBJECT INTEGRATION INTERFACES
ICLOCK
416
414
REHYDRATOR
410
412
CONTRACT
• PACKETS
• REQUEST/ RESPONSE PATTERNS
• PROTOCOLS
413
NETWORK DATA PACKETS
106
CONTROLLED DEVICE (E.G., CLOCK)
226
DESCRIP-TION DOCUMENT
232
CONTROL SERVER

FIG. 9

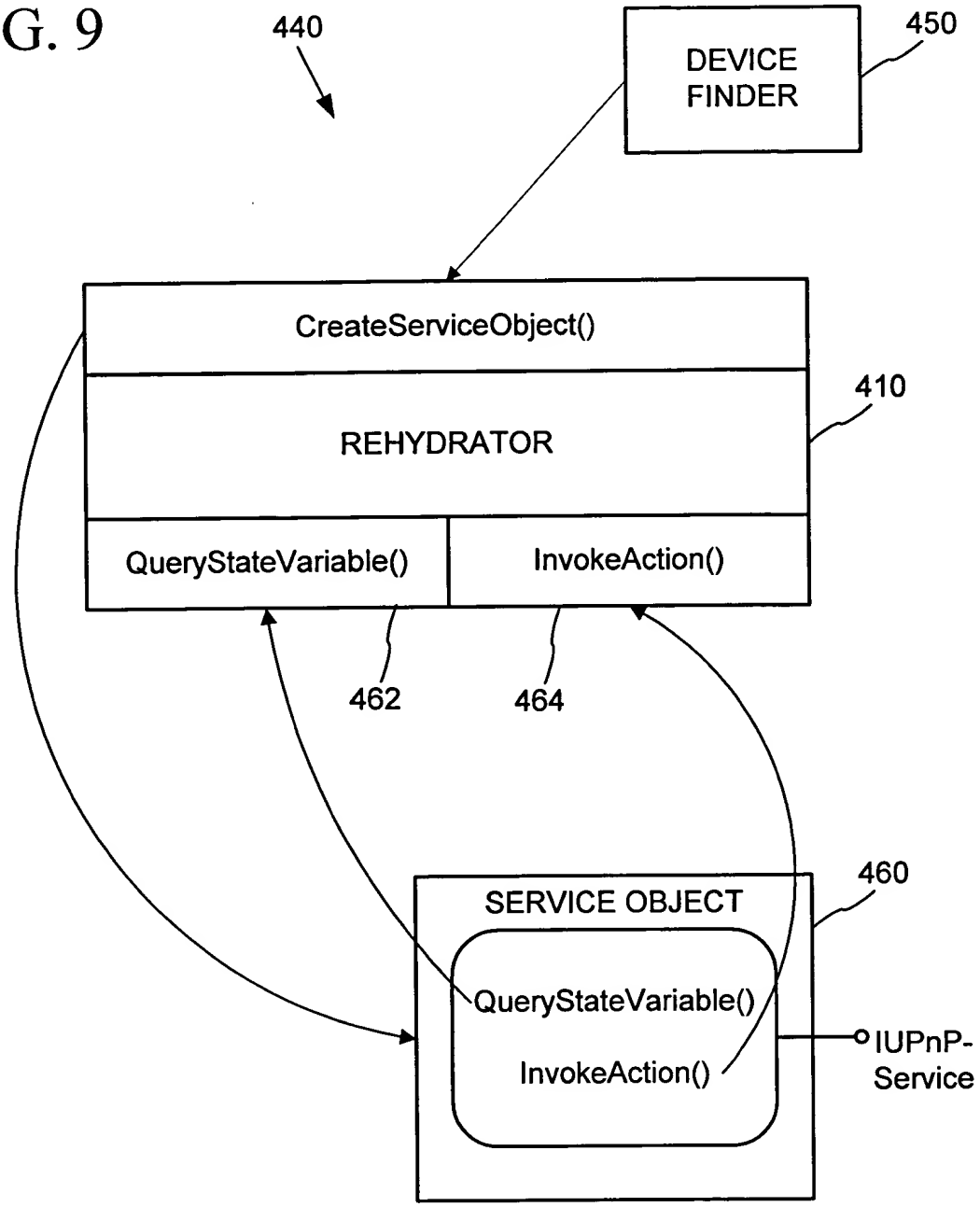
440
DEVICE FINDER
450
CreateServiceObject()
REHYDRATOR
410
QueryStateVariable()
InvokeAction()
462
464
SERVICE OBJECT
460
QueryStateVariable()
InvokeAction()
IUPnP-Service

FIG. 10

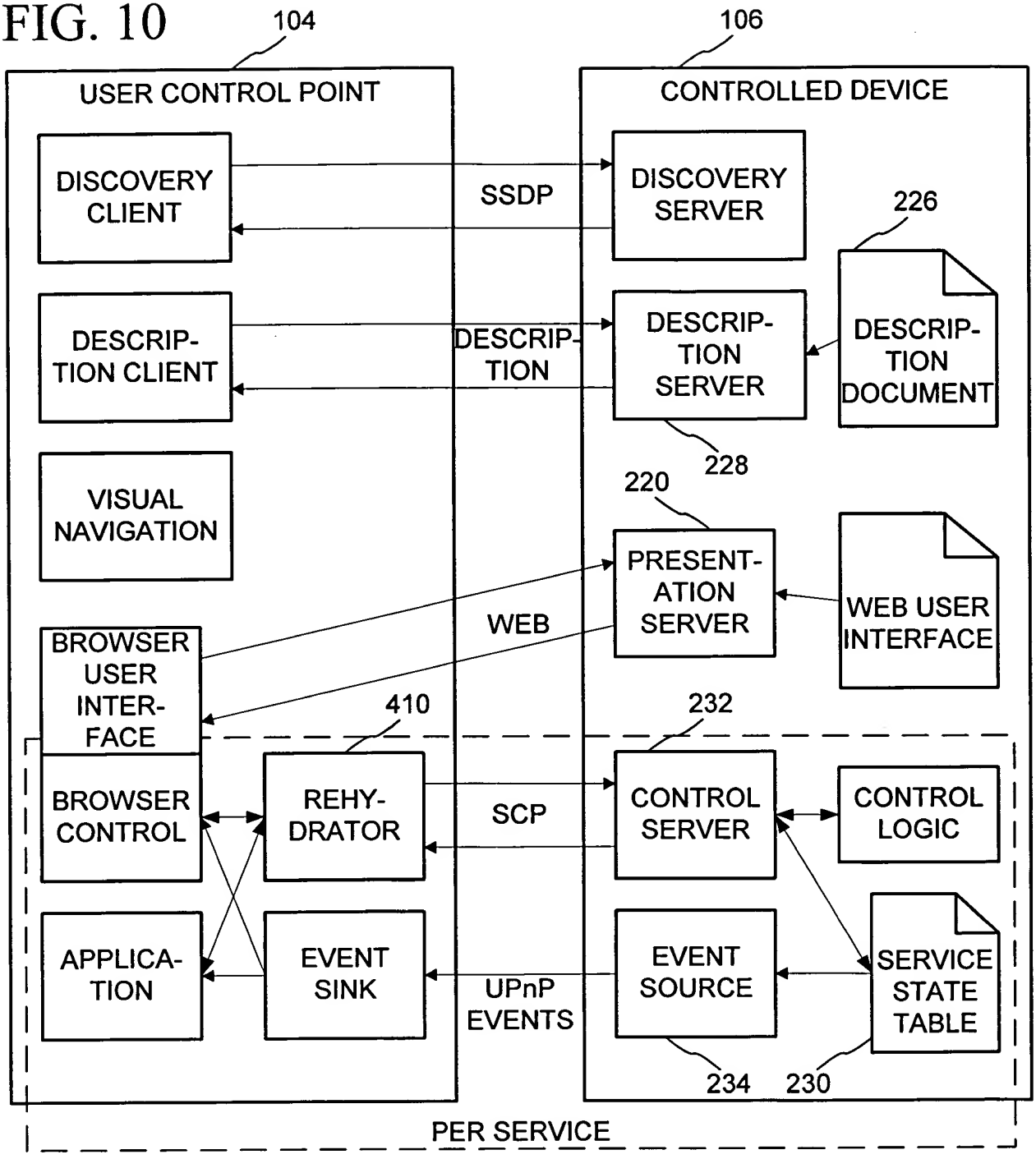
104
106
USER CONTROL POINT
CONTROLLED DEVICE
DISCOVERY CLIENT
SSDP
DISCOVERY SERVER
226
DESCRIP-TION CLIENT
DESCRIP-TION
DESCRIP-TION SERVER
DESCRIP-TION DOCUMENT
228
VISUAL NAVIGATION
220
PRESENT-ATION SERVER
WEB USER INTERFACE
WEB
BROWSER USER INTER-FACE
410
232
BROWSER CONTROL
REHY-DRATOR
SCP
CONTROL SERVER
CONTROL LOGIC
APPLICA-TION
EVENT SINK
UPnP EVENTS
EVENT SOURCE
SERVICE STATE TABLE
234
230
PER SERVICE

FIG. 11

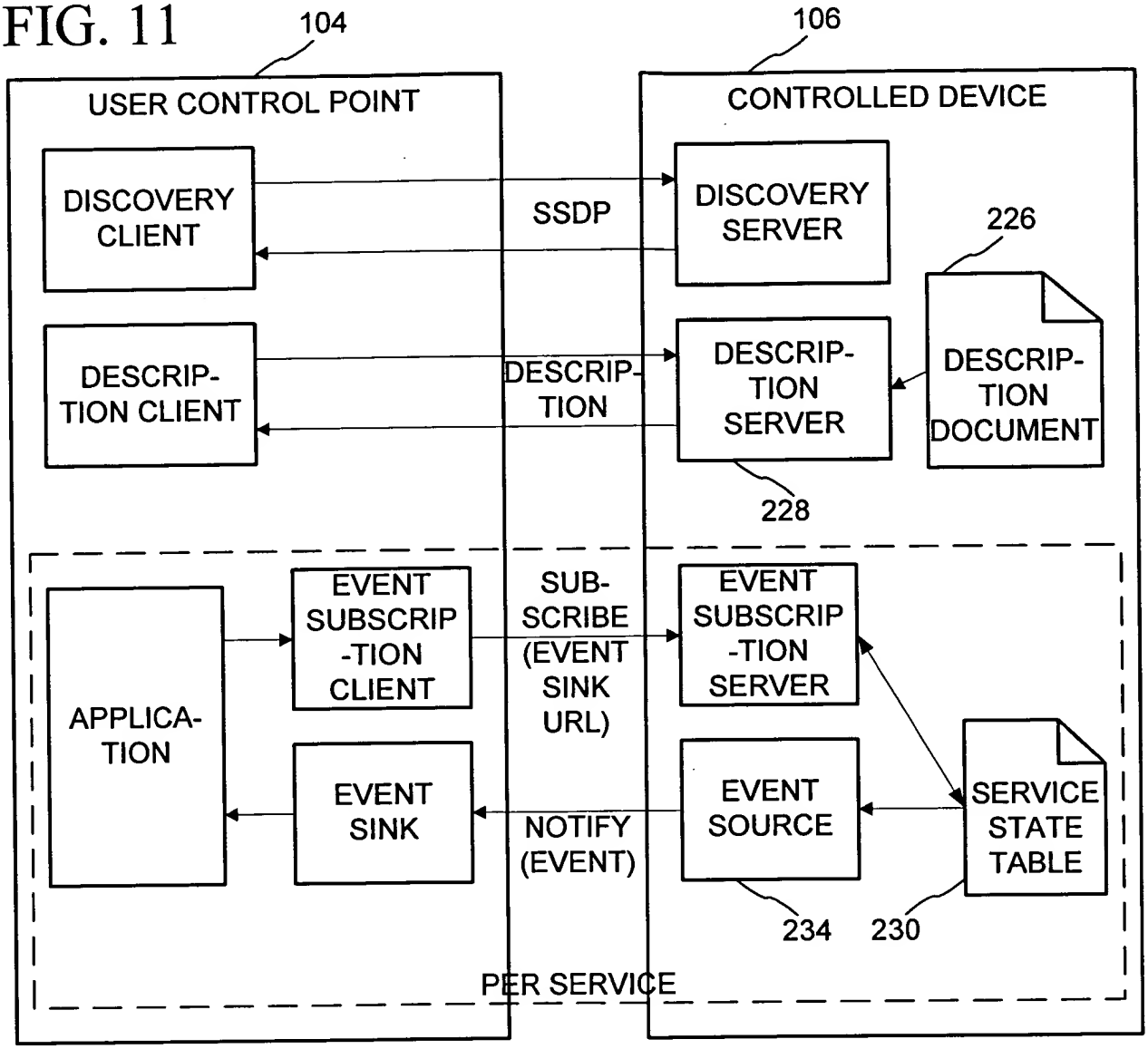
104
106
USER CONTROL POINT
CONTROLLED DEVICE
DISCOVERY CLIENT
SSDP
DISCOVERY SERVER
226
DESCRIP-TION CLIENT
DESCRIP-TION
DESCRIP-TION SERVER
DESCRIP-TION DOCUMENT
228
APPLICA-TION
EVENT SUBSCRIP-TION CLIENT
SUB-SCRIBE (EVENT SINK URL)
EVENT SUBSCRIP-TION SERVER
EVENT SINK
NOTIFY (EVENT)
EVENT SOURCE
SERVICE STATE TABLE
234
230
PER SERVICE

FIG. 12

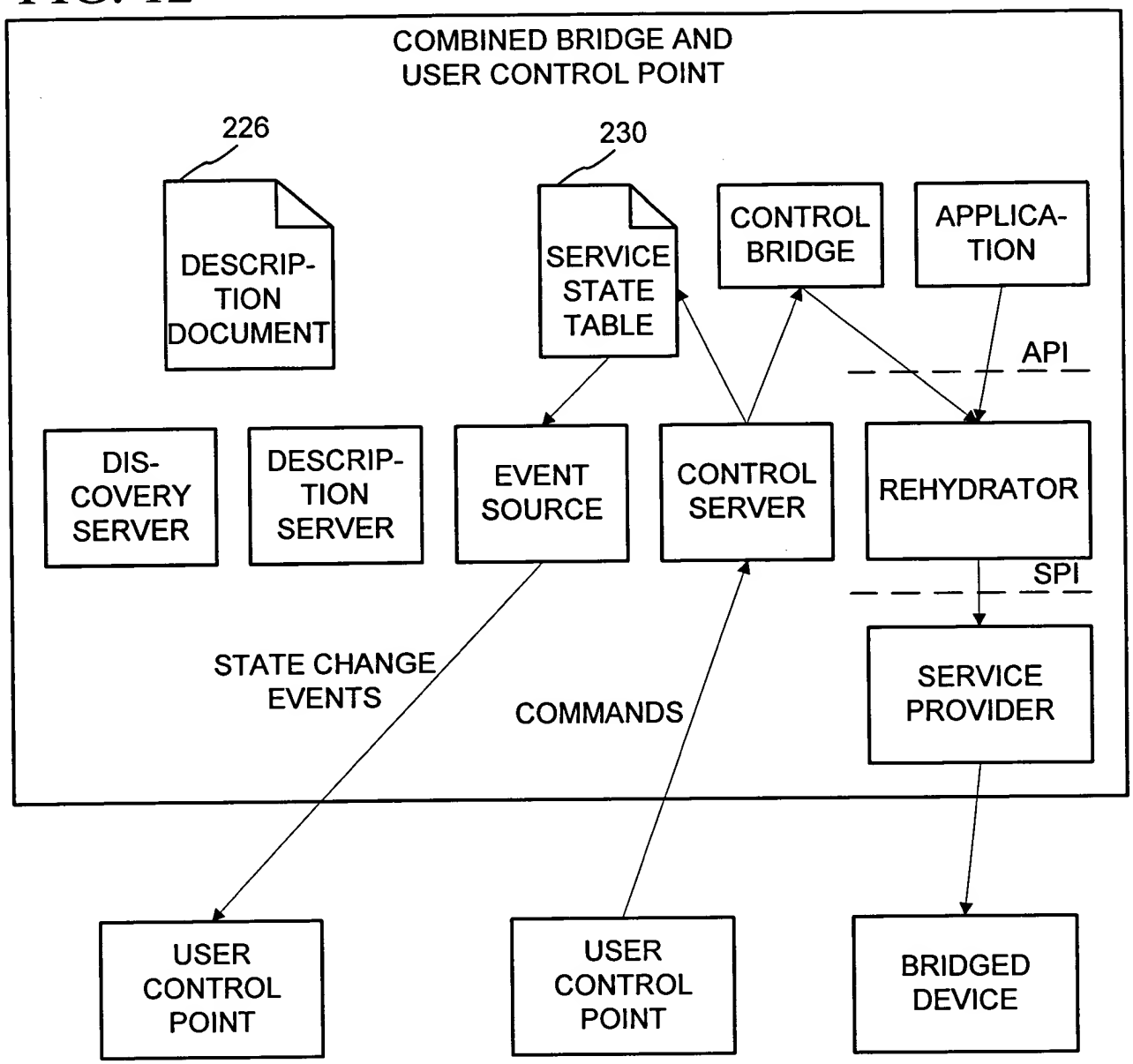
COMBINED BRIDGE AND
USER CONTROL POINT
226
230
DESCRIP-
TION
DOCUMENT
SERVICE
STATE
TABLE
CONTROL
BRIDGE
APPLICA-
TION
API
DIS-
COVERY
SERVER
DESCRIP-
TION
SERVER
EVENT
SOURCE
CONTROL
SERVER
REHYDRATOR
SPI
STATE CHANGE
EVENTS
COMMANDS
SERVICE
PROVIDER
USER
CONTROL
POINT
USER
CONTROL
POINT
BRIDGED
DEVICE

FIG. 13

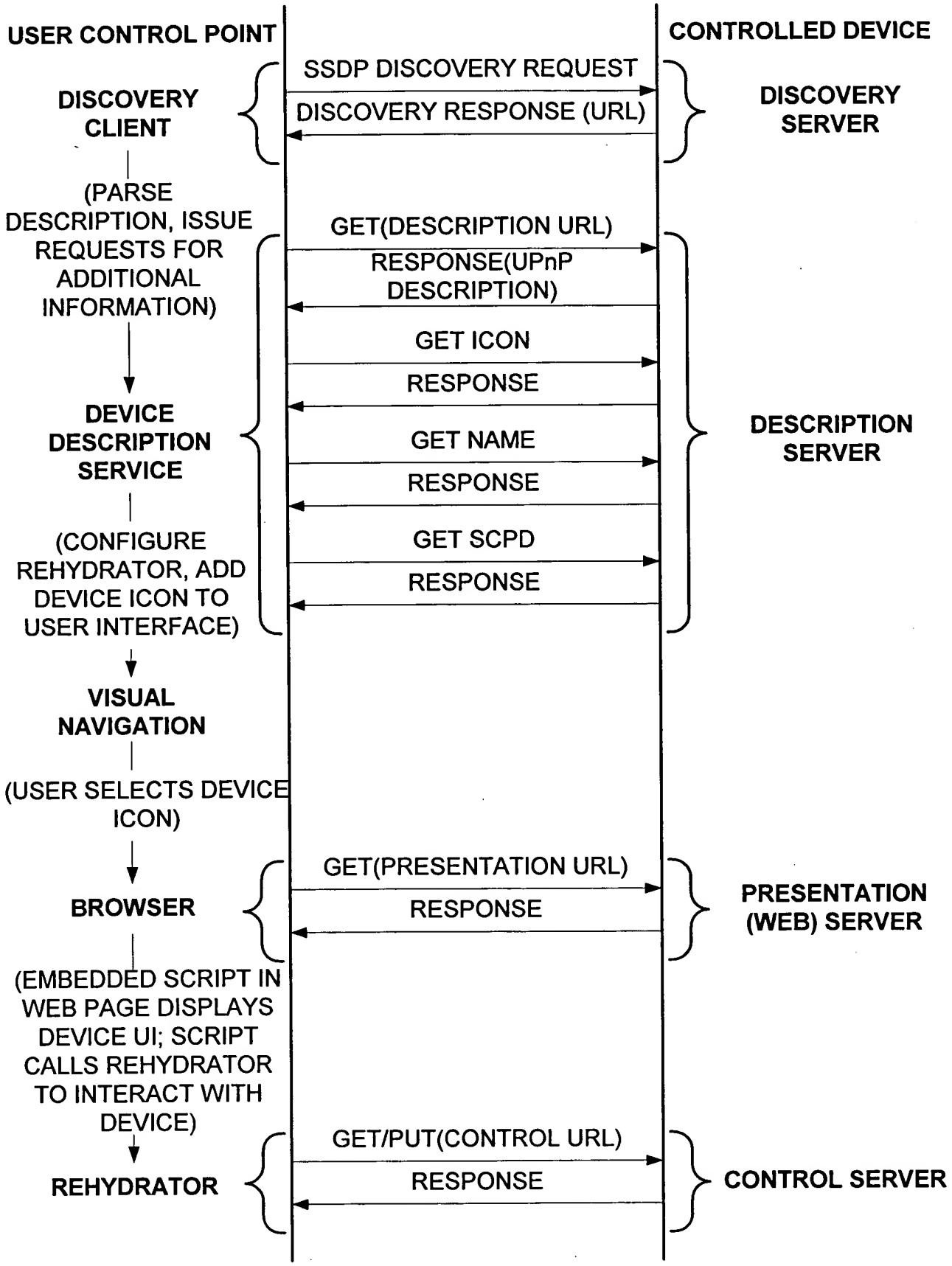
USER CONTROL POINT
CONTROLLED DEVICE
SSDP DISCOVERY REQUEST
DISCOVERY RESPONSE (URL)
DISCOVERY CLIENT
DISCOVERY SERVER
(PARSE DESCRIPTION, ISSUE REQUESTS FOR ADDITIONAL INFORMATION)
GET(DESCRIPTION URL)
RESPONSE(UPnP DESCRIPTION)
GET ICON
RESPONSE
GET NAME
RESPONSE
GET SCPD
RESPONSE
DEVICE DESCRIPTION SERVICE
DESCRIPTION SERVER
(CONFIGURE REHYDRATOR, ADD DEVICE ICON TO USER INTERFACE)
VISUAL NAVIGATION
(USER SELECTS DEVICE ICON)
GET(PRESENTATION URL)
RESPONSE
BROWSER
PRESENTATION (WEB) SERVER
(EMBEDDED SCRIPT IN WEB PAGE DISPLAYS DEVICE UI; SCRIPT CALLS REHYDRATOR TO INTERACT WITH DEVICE)
GET/PUT(CONTROL URL)
RESPONSE
REHYDRATOR
CONTROL SERVER

FIG. 14

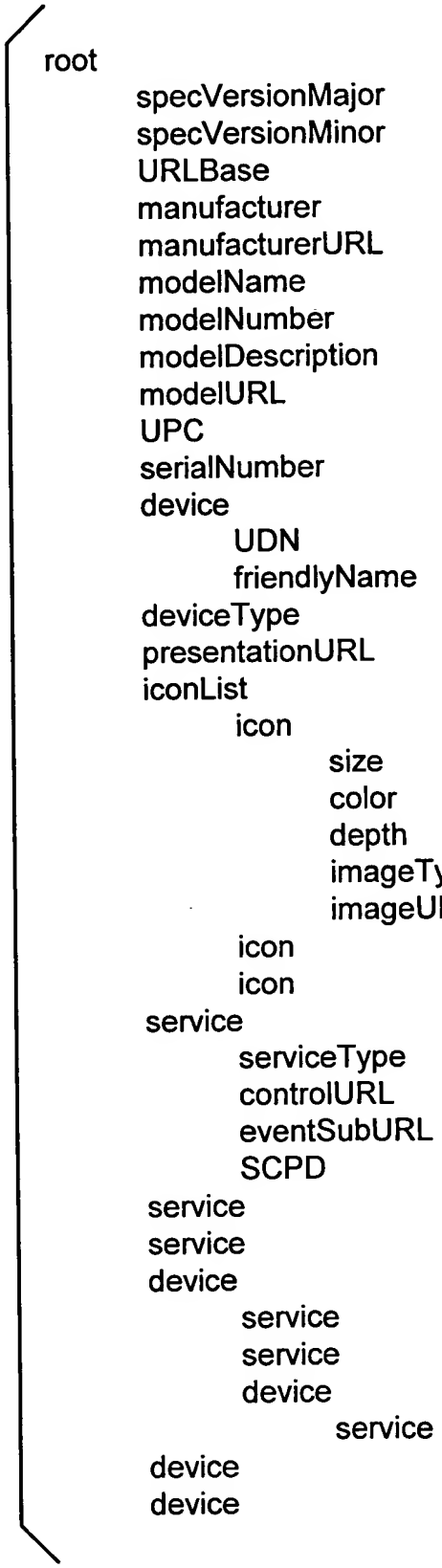

FIG. 15

```
<device>
  ...
  <iconList>
    <icon>
      <size>16</size>
      <color>0</color>
      <depth>8</depth>
      <imageType>PNG</imageType>
      <image>"http://device.local/iconpath/icon16bw.png"</image>
    </icon>
    <icon>
      <size>32</size>
      <color>0</color>
      <depth>8</depth>
      <imageType>PNG</imageType>
      <image>"http://device.local/iconpath/icon32bw.png"</image>
    </icon>
    <icon>
      <size>48</size>
      <color>0</color>
      <depth>8</depth>
      <imageType>PNG</imageType>
      <image>"http://device.local/iconpath/icon48bw.png"</image>
    </icon>
    <icon>
      <size>16</size>
      <color>1</color>
      <depth>8</depth>
      <imageType>PNG</imageType>
      <image>"http://device.local/iconpath/icon16c.png"</image>
    </icon><device>
    <icon>
      <size>32</size>
      <color>0</color>
      <depth>8</depth>
      <imageType>PNG</imageType>
      <image>"http://device.local/iconpath/icon32c.png"</image>
    </icon>
    <icon>
      <size>48</size>
      <color>0</color>
      <depth>8</depth>
      <imageType>PNG</imageType>
      <image>"http://device.local/iconpath/icon48c.png"</image>
    </icon>
    ...
  </iconList>
  ...
</device>
```

FIG. 16

```
<?xml version="1.0"?>
<scpd xmlns="x-schema:scpdl-schema.xml">
   <service StateTable>
      <stateVariable>
         <name>currentChannel</name>
         <dataType>number</dataType>
         <allowedValueRange>
            <minimum>0</minimum>
            <maximum>55</maximum>
            <step>1</step>
         </allowedValueRange>
      </stateVariable>
   </serviceStateTable>


   <actionList>
      <action>
         <name>ChannelUp</name>
      </action>

      <action>
         <name>ChannelDown</name>
      </action>

      <action>
         <name>SetChannel</name>
         <argument>
            <name>newChannel</name>
  <relatedStateVariable>
currentChannel
  </relatedStateVariable>
         </argument>
      </action>
   </actionList>
</scpd>
```

FIG. 17

```
<contract>
  <protocol id="protocolDef">
    <HTTP version="1.1">
      <URL></URL>
      <M-POST>
        <MAN>http://www.microsoft.com/protocols/ext/XOAP</MAN>
      </M-POST>
      <HEADER name="Content-Type" value="text/xml" />
          <!-- Need to put in extension headers here -->
    </HTTP>


  </protocol>

  <RequestResponse name="queryStateVariable">
    <protocol is="protocolDef">
    <in is="queryStateVariable">
    <out is="queryStateVariableResponse">
    <error is="queryStateVariableResponse">
  </RequestResponse>

  <RequestResponse name="invokeAction">
    <protocol is="protocolDef">
    <in is="SerializedStream">
    <out is="invokeActionResponse">
    <error is="invokeActionResponse">
  </RequestResponse>


  <Schema name="upnp_scpdl"
    xmlns="urn:schemas-microsoft-com:xml-data"
    xmlns:dt="urn:schemas-microsoft-com:datatypes">

    <!-- Common -->

    <ElementType name="_return" content="textOnly" dt:type="string" />
    <ElementType name="_fault" content="textOnly" dt:type="string" />

    <!-- Query State Variable Call -->

    <ElementType name="variableName" content="textOnly" dt:type="string" />

    <ElementType name="queryStateVariable" content="eltOnly" model="closed">
      <element type="variableName" />
    </ElementType>

    <!-- Query State Variable Response -->
...
```

FIG. 18

```
...
      <ElementType name="queryStateVariableResponse" content="eltOnly"
model="closed">
         <group order="one">
            <element type="_return">
            <element type="_fault">
         </group>
      </ElementType>

      <!-- Invoke Action Call -->

      <AttributeType name="main" dt:type="idref" />
      <AttributeType name="headers" dt:type="idref" />
      <AttributeType name="id" dt:type="id" />

      <ElementType name="sequenceNumber" content="textOnly" dt:type="int">
         <AttrbuteType name="dt" dt:type="string" dt:values="int" />

         <attribute type="dt" />
      </ElementType>

      <ElementType name="headers" content="eltOnly" model="closed"
         <attribute type="id" required="yes" />
         <element type="sequenceNumber" />
      </ElementType>

      <ElementType name="actionName" content="textOnly" dt:type="string" />
      <ElementType name="actionArg" content="textOnly" dt:type="string" />

      <ElementType name="invokeAction" content="eltOnly" model="closed">
         <attribute type="id" required="yes" />

         <element type="actionName">
         <element type="actionArg" minOccurs="0" maxOccurs="*">
      </ElementType>
...
```

FIG. 19

```
...
      <ElementType name="SerializedStream" content="eltOnly" model="closed">
         <attribute type="main" required="yes" />
         <attribute type="headers" required="yes" />

         <element type="headers">
         <element type="invokeAction">

      </ElementType>

      <!-- Invoke Action Response -->

      <ElementType name="invokeActionResponse" content="eltOnly" model="closed">
         <group order="one">
            <element type="_return">
            <element type="_fault">
         </group>
      </ElementType>
   </Schema>
</contract>
```

FIG. 20

```
<?xml version="1.0"?>
<Schema name="upnp_scpdl"
   xmlns="urn:schemas-microsoft-com:xml-data"
   xmlns:dt="urn:schemas-microsoft-com:datatypes">

   <!-- Common Elements and Attributes -->

   <ElementType name="name" content="textOnly" dt:type="string" />

   <!-- Service State Table -->

   <ElementType name="minimum" content="textOnly" dt:type="number" />
   <ElementType name="maximum" content="textOnly" dt:type="number" />
   <ElementType name="step" content="textOnly" dt:type="number" />

   <ElementType name="allowedValueRange" content="eltOnly" model="closed">
      <element type="minimum" />
      <element type="maximum" />
      <element type="step" minOccurs="0" />
   </ElementType>

   <ElementType name="allowedValue" content="textOnly" />

   <ElementType name="allowedValueList" content="eltOnly" model="closed">
      <element type="allowedValue" minOccurs="1" maxOccurs="*" />
   </ElementType>

   <ElementType name="dataType" content="textOnly" dt:type="string" />

   <ElementType name="stateVariable" content="eltOnly" model="closed">
      <element type="name" />
...
```

FIG. 21

```
...
      <element type="dataType" />

      <group minOccurs="0" maxOccurs="1" order="one">
        <element type="allowedValueRange" />
        <element type="allowedValueList" />
      </group>
    </ElementType>


    <ElementType name="deviceStateTable" content="eltOnly" model="closed">
      <element type="stateVariable" minOccurs="1" maxOccurs="*" />
    </ElementType>


    <!-- Action List -->

    <ElementType name="relatedStateVariable" content="textOnly" dt:type="string" />

    <ElementType name="argument" content="eltOnly" model="closed">
      <element type="name" />
      <element type="relatedStateVariable" />
    </ElementType>

    <ElementType name="action" content="eltOnly" model="closed">
      <element type="name" />
      <element type="argument" minOccurs="0" maxOccurs="*" />
    </ElementType>

    <ElementType name="actionList" content="eltOnly" model="closed">
      <element type="action" minOccurs="0" maxOccurs="*" />
    </ElementType>

    <!-- Root Element -->

    <ElementType name="dcpd" content="eltOnly" model="closed">
      <element type="deviceStateTable" />
      <element type="actionList" />
    </ElementType>
  </Schema>
```

FIG. 22

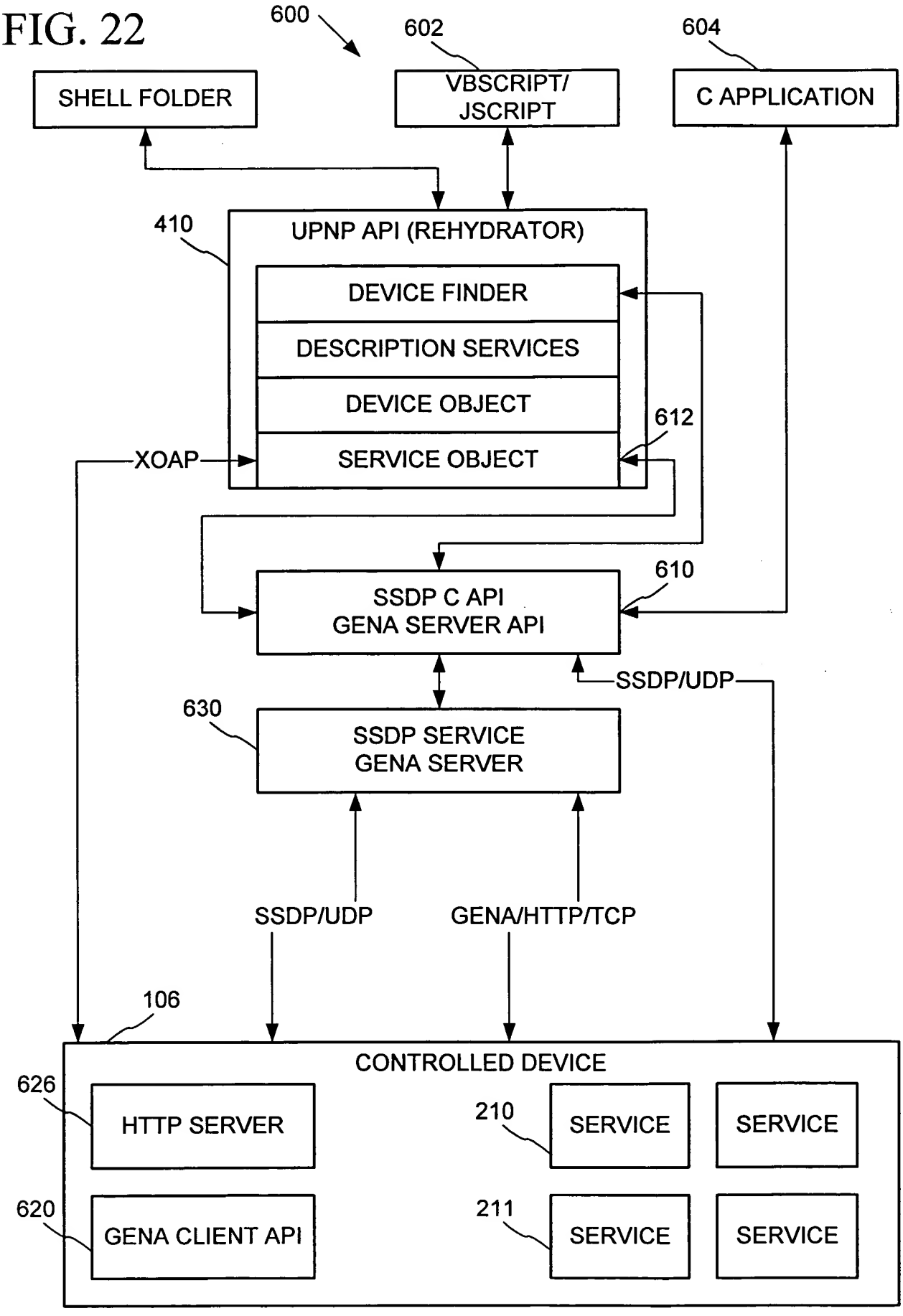
600
602
604
SHELL FOLDER
VBSCRIPT/
JSCRIPT
C APPLICATION
410
UPNP API (REHYDRATOR)
DEVICE FINDER
DESCRIPTION SERVICES
DEVICE OBJECT
612
XOAP
SERVICE OBJECT
610
SSDP C API
GENA SERVER API
SSDP/UDP
630
SSDP SERVICE
GENA SERVER
SSDP/UDP
GENA/HTTP/TCP
106
CONTROLLED DEVICE
626
HTTP SERVER
210
SERVICE
SERVICE
620
GENA CLIENT API
211
SERVICE
SERVICE

FIG. 23

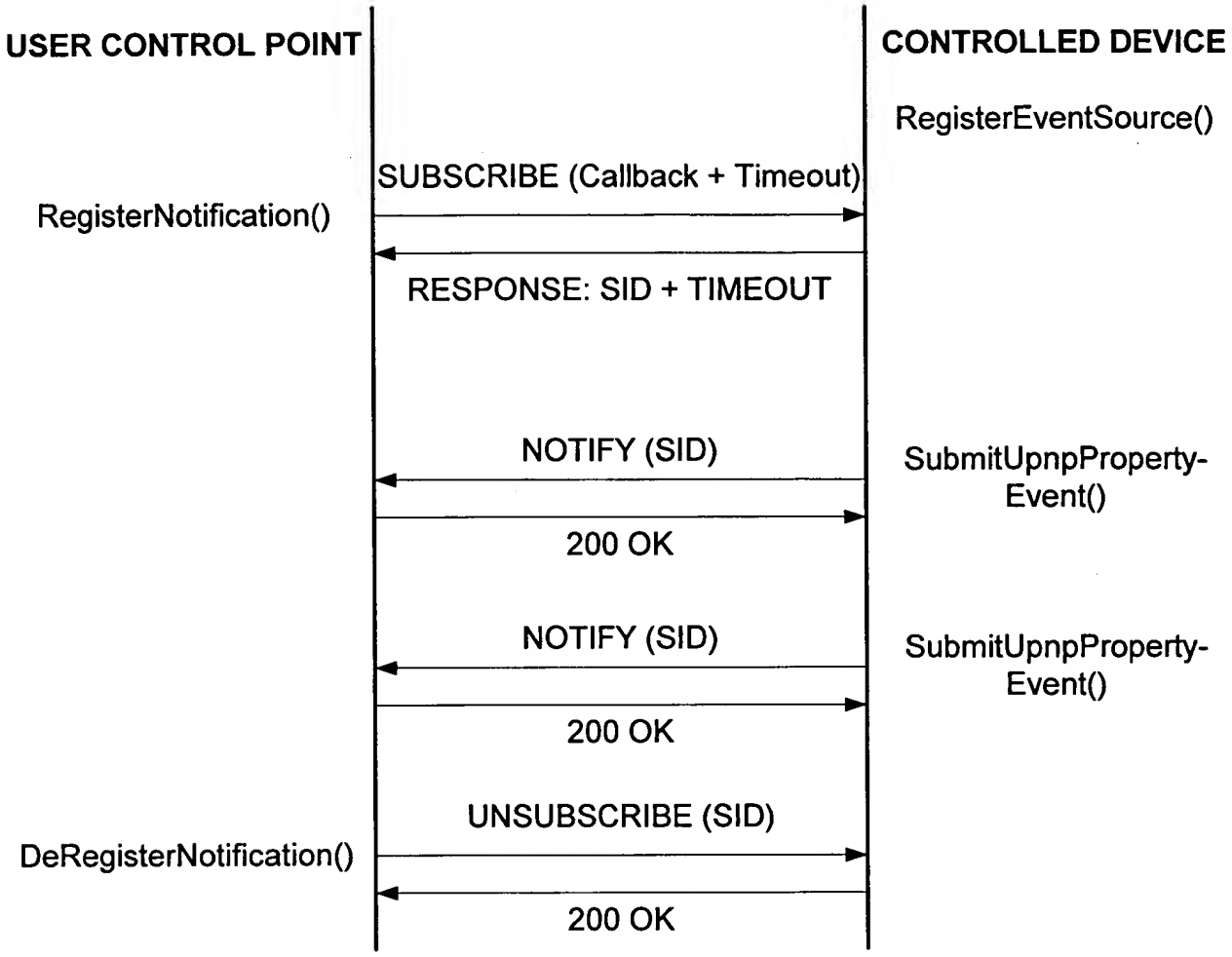
USER CONTROL POINT
CONTROLLED DEVICE
RegisterEventSource()
SUBSCRIBE (Callback + Timeout)
RegisterNotification()
RESPONSE: SID + TIMEOUT
NOTIFY (SID)
SubmitUpnpProperty-Event()
200 OK
NOTIFY (SID)
SubmitUpnpProperty-Event()
200 OK
UNSUBSCRIBE (SID)
DeRegisterNotification()
200 OK

FIG. 24

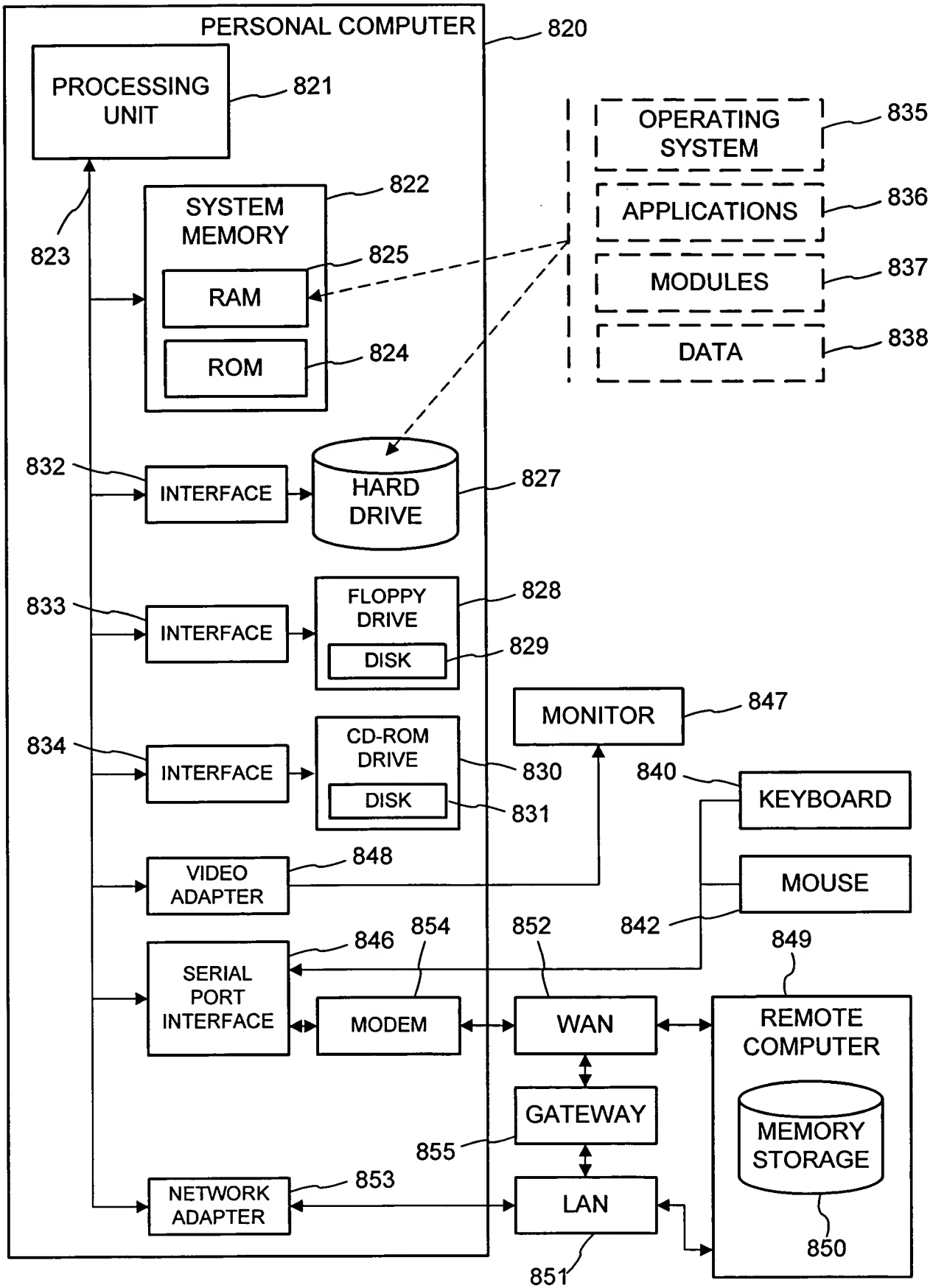
PERSONAL COMPUTER
820
PROCESSING UNIT
821
SYSTEM MEMORY
822
823
RAM
825
ROM
824
OPERATING SYSTEM
835
APPLICATIONS
836
MODULES
837
DATA
838
832
INTERFACE
HARD DRIVE
827
833
INTERFACE
FLOPPY DRIVE
828
DISK
829
834
INTERFACE
CD-ROM DRIVE
830
DISK
831
MONITOR
847
840
KEYBOARD
MOUSE
842
VIDEO ADAPTER
848
SERIAL PORT INTERFACE
846
854
852
849
MODEM
WAN
REMOTE COMPUTER
GATEWAY
855
MEMORY STORAGE
NETWORK ADAPTER
853
LAN
850
851

FIG. 25

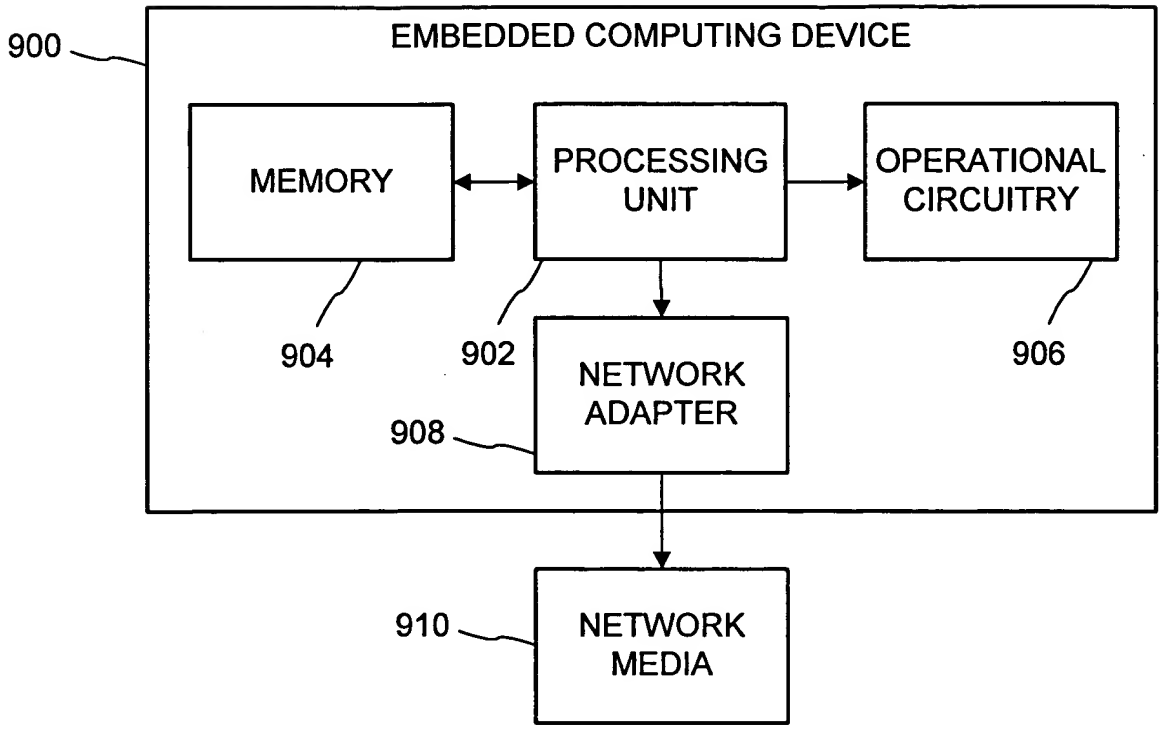
900
EMBEDDED COMPUTING DEVICE
MEMORY
PROCESSING UNIT
OPERATIONAL CIRCUITRY
904
902
906
NETWORK ADAPTER
908
910
NETWORK MEDIA

FIG. 26

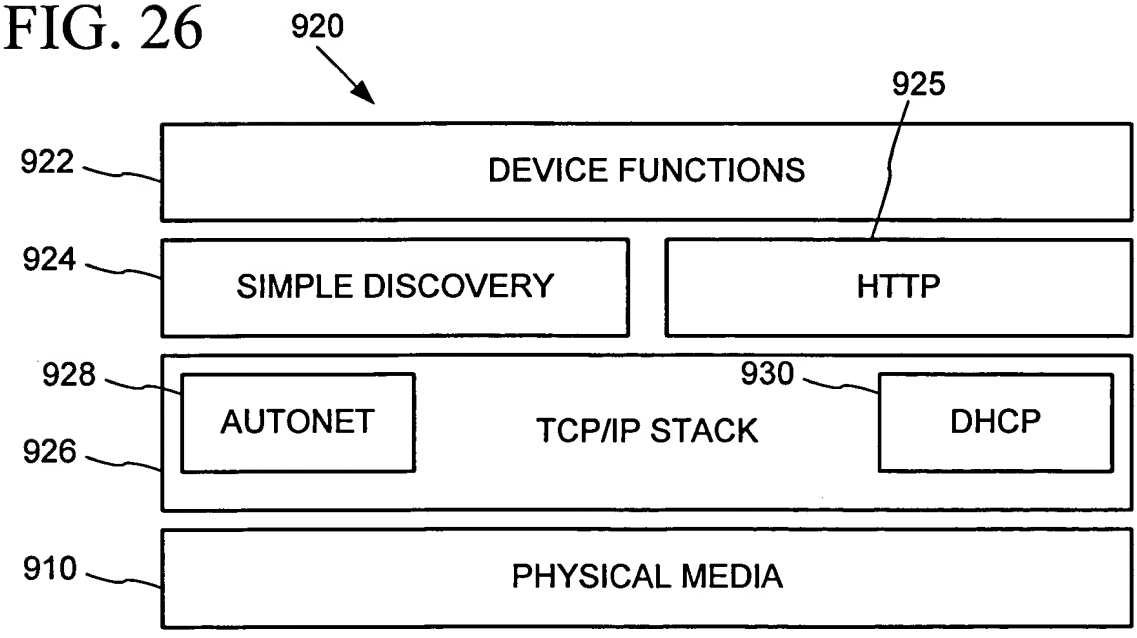
920
925
922
DEVICE FUNCTIONS
924
SIMPLE DISCOVERY
HTTP
928
930
AUTONET
TCP/IP STACK
DHCP
926
910
PHYSICAL MEDIA

FIG. 27

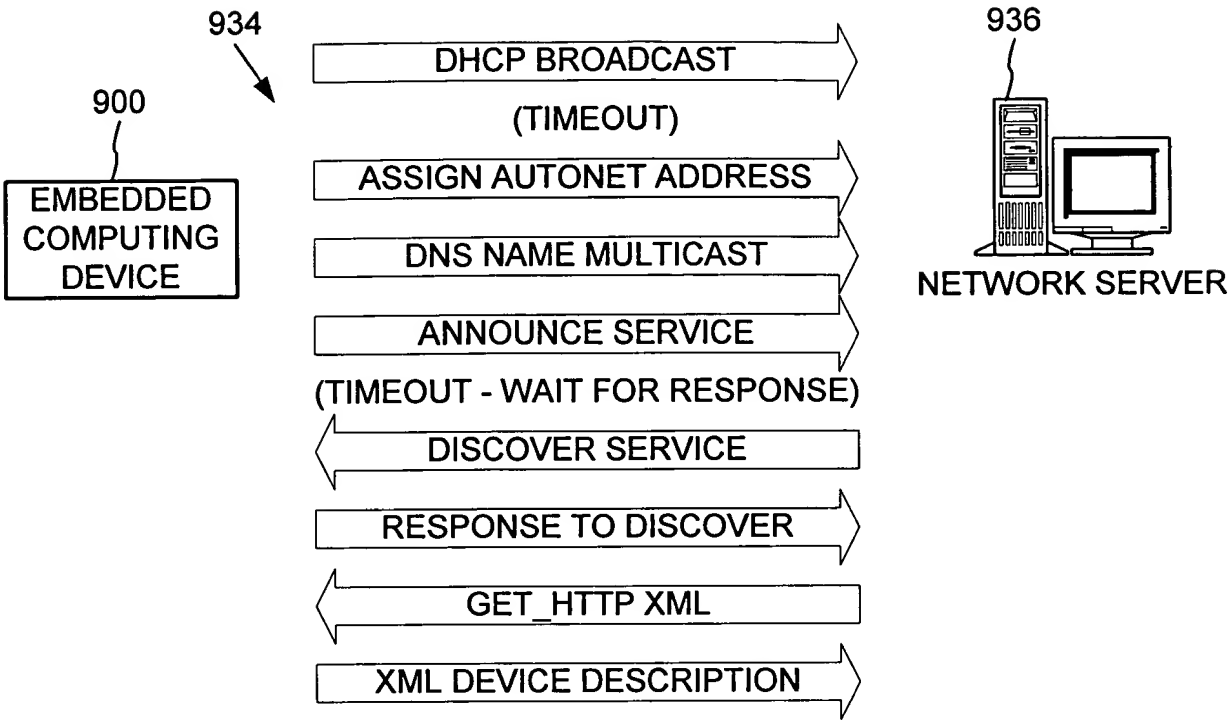
934
900
EMBEDDED COMPUTING DEVICE
DHCP BROADCAST
(TIMEOUT)
ASSIGN AUTONET ADDRESS
DNS NAME MULTICAST
ANNOUNCE SERVICE
(TIMEOUT - WAIT FOR RESPONSE)
DISCOVER SERVICE
RESPONSE TO DISCOVER
GET_HTTP XML
XML DEVICE DESCRIPTION
936
NETWORK SERVER

FIG. 28

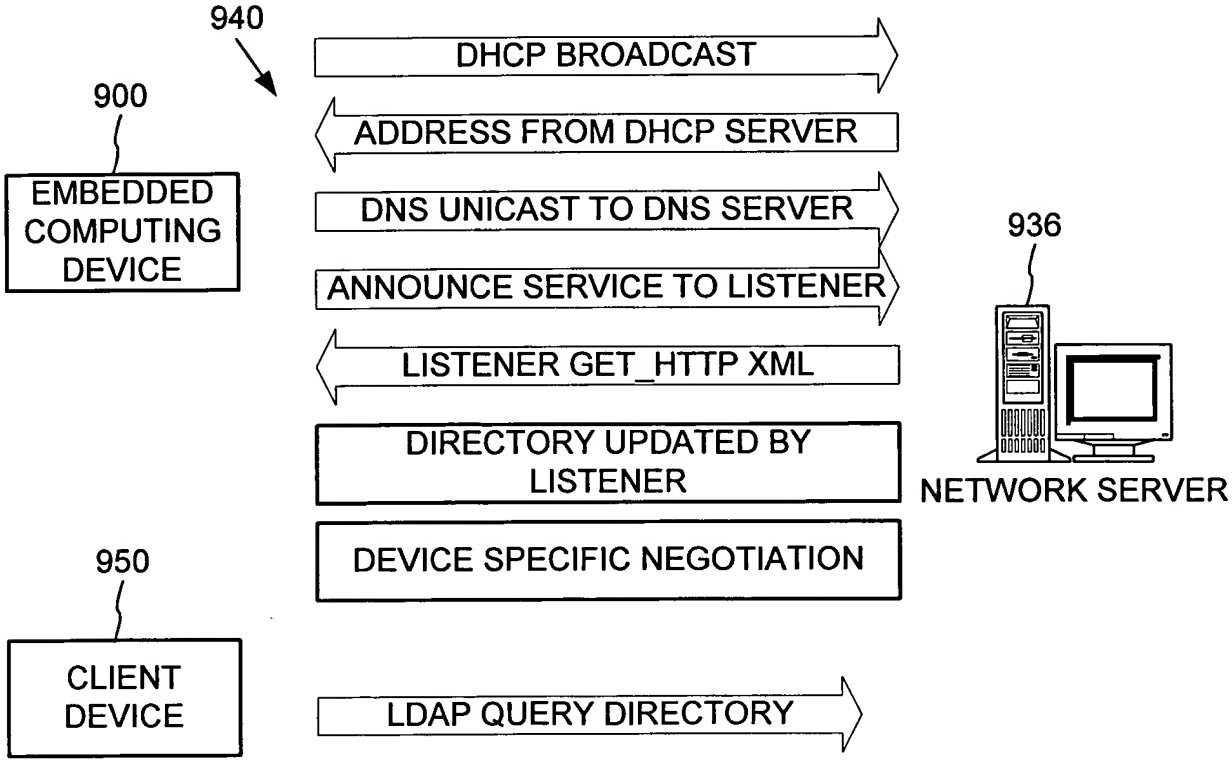
940
900
EMBEDDED COMPUTING DEVICE
DHCP BROADCAST
ADDRESS FROM DHCP SERVER
DNS UNICAST TO DNS SERVER
ANNOUNCE SERVICE TO LISTENER
LISTENER GET_HTTP XML
DIRECTORY UPDATED BY LISTENER
DEVICE SPECIFIC NEGOTIATION
936
NETWORK SERVER
950
CLIENT DEVICE
LDAP QUERY DIRECTORY

FIG. 29

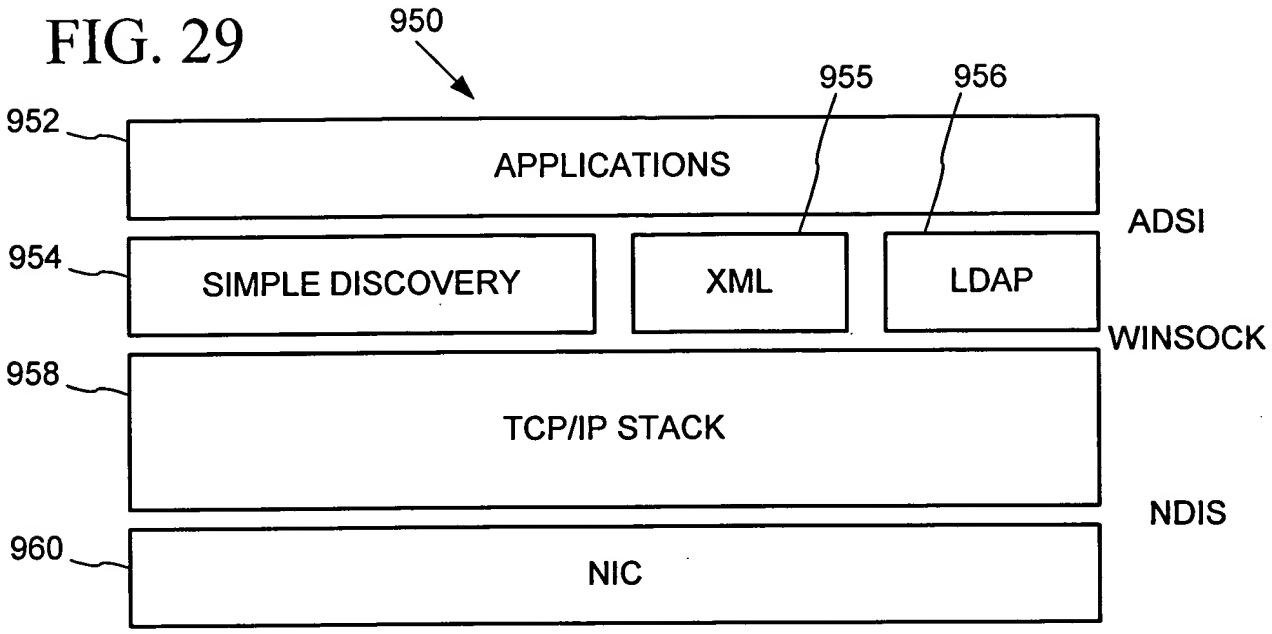

FIG. 30

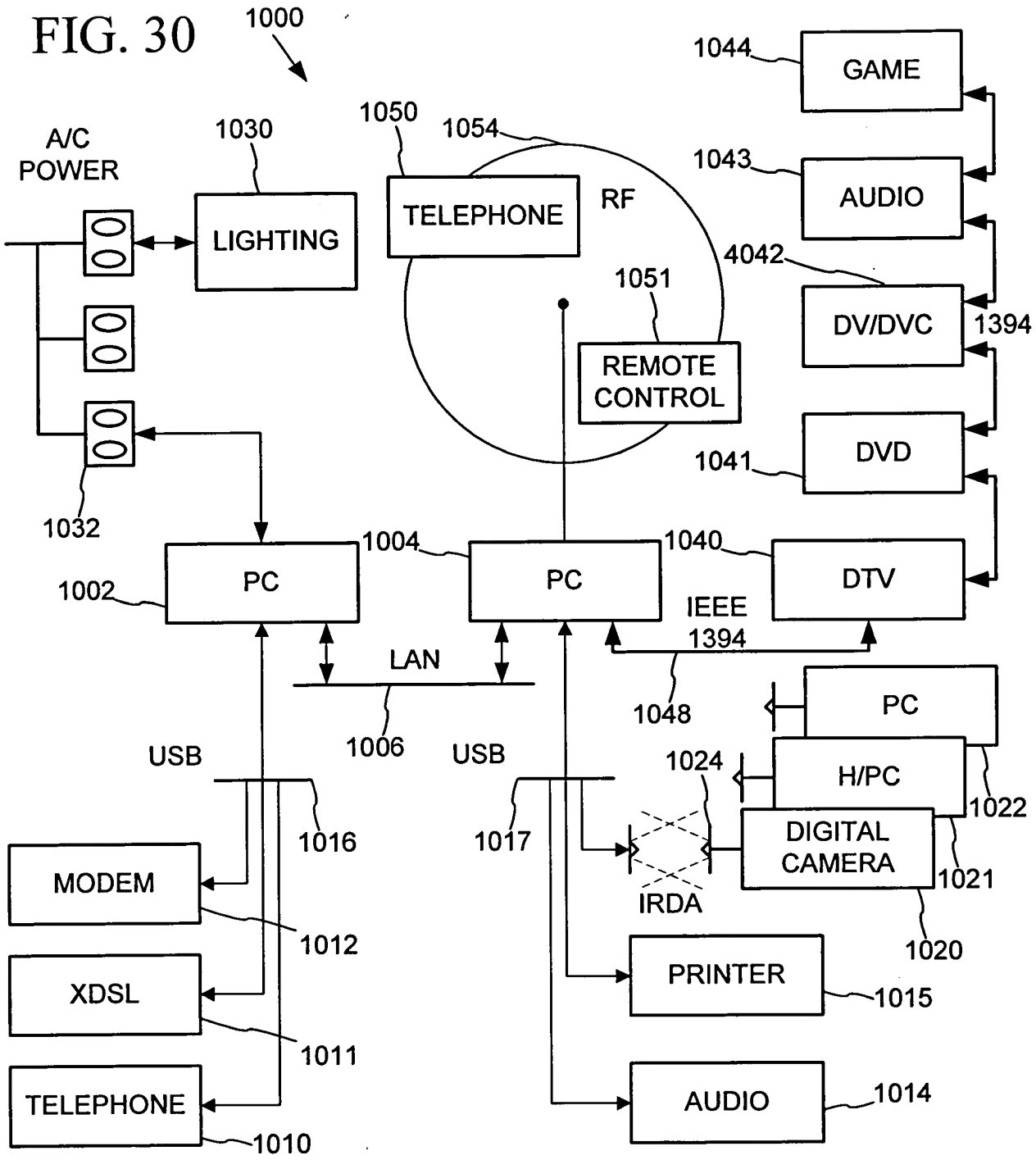
1000
1044
GAME
1050
1054
1030
A/C
POWER
1043
AUDIO
TELEPHONE
RF
LIGHTING
4042
1051
DV/DVC
1394
REMOTE
CONTROL
1041
DVD
1032
1004
1040
1002
PC
PC
DTV
IEEE
1394
LAN
1048
PC
1006
USB
USB
1024
H/PC
1022
1016
1017
DIGITAL
CAMERA
1021
MODEM
IRDA
1020
1012
PRINTER
1015
XDSL
1011
AUDIO
1014
TELEPHONE
1010

FIG. 31

```
[
object,
uuid(<foo>),
dual,
helpstring("IUPNPDevice interface"),
pointer_default(unique)
]
   interface IUPNPDevice : IDispatch
{

   [propget, id(DISPID_UPNPDEVICE_DESCRIPTIONDOCUMENT),
    helpstring("returns the document from which the properties of this device are
being read")]
   HRESULT DescriptionDocument([restricted, hidden, out, retval]
IUPNPDescriptionDocument ** ppuddDocument);
        purpose: returns the document from which the properties of this device are
being read.
        parameters: ppuddDocument, A reference to the description document
object from which data about the device is being read.  This must be freed when no
longer needed.
        return values: S_OK, ppuddDocument is a refernce to the device's
description document.

   [propget, id(DISPID_UPNPDEVICE_ISROOTDEVICE),
    helpstring("denotes whether the physical location information of this device can
be set")]
   HRESULT IsRootDevice([out, retval] VARIANT_BOOL * pvarb);
        parameters: pvarb, the address of a VARIANT_BOOL that will receive the
value of VARIANT_TRUE if the current device is the topmost device in the device
tree, and will receive the value of VARIANT_FALSE otherwise.
        return values: S_OK, varb is set to the appropriate value
        note: if a device is a root device, calls RootDevice() or ParentDevice() will
return NULL


   [propget, id(DISPID_UPNPDEVICE_ROOT),
    helpstring("returns the top device in the device tree")]
   HRESULT RootDevice([out, retval] IUPNPDevice ** ppudDeviceRoot);
        purpose: returns the top device in the device tree
...
```

FIG. 32

...

parameters: ppudDeviceRoot, On return, this refers to the "root" device of the current device tree. The root device is the topmost parent of the current device. If the current device is the root device this method will set *ppudDeviceRoot to null, and return S_FALSE.

return values: S_OK, *ppudDeviceRoot contains a reference to the root device. S_FALSE, the current device is the root device. *ppudDeviceRoot is null.

[propget, id(DISPID_UPNPDEVICE_PARENT),
helpstring("returns the parent of the current device")]
HRESULT ParentDevice([out, retval] IUPNPDevice ** ppudDeviceParent);

parameters: ppudDeviceParent, On return, if the device has a parent, this is the address of a IUPNPDevice object which can describe the parent. This must be released when no longer needed. If the device has no parent (it is a "root" device), than this value will be set to null.

return values: S_OK, ppudDeviceParent contains a reference to the device's parent. S_FALSE, the current device is the root device, which has no parent. *ppudDeviceRoot is null.

[propget, id(DISPID_UPNPDEVICE_CHILDREN),
helpstring("returns a collection of the children of the current device")]
HRESULT Children([out, retval] IUPNPDevices ** ppudChildren);

parameters: ppudChildren, On return, this is the address of a newly-created IUPNPDevices collection that can enumerate this device's children. This must be released when no longer needed. If the device has no children, this method will return a collection object with a length of zero.

return values: S_OK, ppudChildren contains a list of the device's children.

[propget, id(DISPID_UPNPDEVICE_UDN),
helpstring("returns the UDN of the device")]
HRESULT UniqueDeviceName([out, retval] BSTR * pbstrUDN);

parameters: pbstrUDN, On return, this contains the address of a newly-allocated string which contains the device's Unique Device Name (UDN). The UDN is globally unique across all devices - no two devices will ever have the same UDN. This value must be freed when no longer needed.

return values: S_OK pbstrUDN contains the UDN of the device

...

FIG. 33

...
[propget, id(DISPID_UPNPDEVICE_DISPLAYNAME),
helpstring("returns the (optional) display name of the device")]
HRESULT DisplayName([out, retval] BSTR * pbstrDisplayName);
parameters: pbstrDisplayName, On return, this contains the address of the device's display name. This value must be freed when no longer needed. If the device does not specify a display name, this parameter will be set to null.
return values: S_OK, bstrDisplayName contains the display name of the device. pbstrDisplayName must be freed. S_FALSE, the device did not specify a display name. *pbstrDisplayName is set to null.
note: it is possible for multiple devices to have the same display name. Applications should use UniqueDeviceName() to determine if two device objects refer to the same device.

[propget, id(DISPID_UPNPDEVICE_CANSETDISPLAYNAME),
helpstring("denotes whether the physical location information of this device can be set")]
HRESULT CanSetDisplayName([out, retval] VARIANT_BOOL * pvarb);
parameters: pvarb, the address of a VARIANT_BOOL. This is true (!=0) on return when the device's display name can be set (via SetDisplayName)
return values: S_OK varb is set to the appropriate value

[id(DISPID_UPNPDEVICE_SETDISPLAYNAME),
helpstring("sets the display name on the device")]
HRESULT SetDisplayName([in] BSTR bstrDisplayName);
parameters: bstrDisplayName, the value to set the device's display name to.
return values: S_OK, varb is set to the appropriate value.
note: On success, this method sets the display name used by a device. Note that this method changes the display name on the device itself, not simply on the local object. This will block while the name is being set.
Additionally, this change will be made on the device alone, and will not be reflected in the current device object. After a successful call to this method, DisplayName will continue to return the 'old' value). To read the device's current name, the caller must re-load the device's description.

[propget, id(DISPID_UPNPDEVICE_DEVICETYPE),
...

FIG. 34

```
...
   helpstring("returns the device type URI")]
   HRESULT Type([out, retval] BSTR * pbstrType);
        parameters: pbstrType, On return, this contains the address of a newly-allocated
string containing the device's type URI. This value must be freed when no longer
needed.
        return values: S_OK, bstrType contains the type URI of the device, and must be
freed when no longer needed.

   [propget, id(DISPID_UPNPDEVICE_SERVICES),
    helpstring("returns the collection of services exposed by the device")]
   HRESULT Services([out, retval] IUPNPServices ** ppusServices);
        parameters: ppusServices, On return, this is the address of a newly-created
IUPNPServices collection that can enumerate the services exposed by the device.
This must be released when no longer needed. If the device exposes no services, this
method will return a collection object with a length of zero.
        return values: S_OK, pusServices contains a list of the device's children.

   [propget, id(DISPID_UPNPDEVICE_SERVICEIDENTIFIER),
    helpstring("returns the (optional) service identifier of the device")]
   HRESULT ServiceIdentifier([out, retval] BSTR * pbstrServiceID );
        parameters: pbstrServiceID, On return, this contains the address of a newly-
allocated string containing the contents of the device's ServiceIdentifier element, if the
device specifies one. This value must be freed when no longer needed. If the device
does not specify a ServiceIdentifier value, this parameter will be set to null.
        return value: S_OK, bstrServiceID contains the service identifier of the device.
pbstrServiceID must be freed. S_FALSE, the device did not specify a service identifier.
*pbstrServiceID is set to null.
        note having a ServiceIdentifier is mutually exclusive with having services. Any
device will either have a list of services or a ServiceIdentifier, but not both.

   [id(DISPID_UPNPDEVICEDESCRIPTION_LOADSMALLICON),
    helpstring("loads a small (titlebar-sized) icon representing the device, encoded in the
specified format")]
   HRESULT LoadSmallIcon([in] BSTR bstrEncodingFormat,
    [out, retval] BSTR * pbstrIconURL);
        parameters:
...
```

FIG. 35

...
bstrEncodingFormat, A string containing the mime-type representing the desired encoding format of the icon. pbstrIconURL, On return, *pbstrIconURL contains a newly-allocated string representing the URL from which the icon can be loaded. This string must be freed when no longer needed.
return values: S_OK, *pbstrIconURL contains a reference to an icon, encoded in the desired encoding format.

[id(DISPID_UPNPDEVICEDESCRIPTION_LOADICON),
helpstring("loads a standard-sized icon representing the device, encoded in the specified format")]
HRESULT LoadIcon([in] BSTR bstrEncodingFormat,
[out, retval] BSTR * pbstrIconURL);
parameters: bstrEncodingFormat, A string containing the mime-type representing the desired encoding format of the icon. pbstrIconURL, On return, *pbstrIconURL contains a newly-allocated string representing the URL from which the icon can be loaded. This string must be freed when no longer needed.
return values: S_OK, *pbstrIconURL contains a reference to an icon, encoded in the desired encoding format.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_PRESENTATIONURL),
helpstring("obtains a presentation URL to a web page that can control the device")]
HRESULT PresentationURL([out, retval] BSTR * pbstrURL);
parameters: pbstrURL, on return, the address of a newly-allocated string containing the web-page-based control URL. If the device did not specify a presentation URL, an empy string ("") will be returned.
return values:S_OK, bstrURL contains a newly-allocated URL that must be freed when no longer needed. S_FALSE, the device does not have a presentation URL. pbstrURL is set to null.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_PHYSICALLOCATION),
helpstring("a set of properties describing the device's physical location")]
HRESULT PhysicalLocation([out, retval] IUPNPPropertyBag * pupl);
parameters: pupl on return, the address of a newly-allocated UPNPPropertyBag object which contains information about the device's physical location
return values
...

FIG. 36

...

S_OK upl contains a newly-allocated object that the caller must free when it is no longer needed.

note: if the object does not provide any description information, an empy property bag will be returned. See SetPhysicalLocation for a listing of defined values in a physical location property bag.

[propget,
id(DISPID_UPNPDEVICEDESCRIPTION_CANSETPHYSICALLOCATION),
helpstring("denotes whether the physical location information of this device can be set")]
HRESULT CanSetPhysicalLocation([out, retval] VARIANT_BOOL * pvarb);

parameters: pvarb the address of a VARIANT_BOOL. This is true (!=0) on return when the device's physical location can be set (via SetPhysicalLocation)

return values: S_OK varb is set to the appropriate value

[id(DISPID_UPNPDEVICEDESCRIPTION_SETPHYSICALLOCATION),
helpstring("writes a set of properties describing the device's physical location to the device")]
HRESULT SetPhysicalLocation([in] IUPNPPropertyBag * pupl);

parameters: pupl A UPNPPropertyBag object which contains the name-value pairs representing the device's current location. the function will not free the object.

return values: S_OK he device has been updated with the supplied physical location information

note: the following are standard values in the physical location property bag: country, campus, building, floor, wing, room, latitude, longitude, altitude. These values can be used programmatically to implement sorting or filtering functionality based on the device's location. Additionally the property bag supports the following value: description, which contains a user-displayable string representing a device's location which does not have programattic significance. Additionally, the physical location update will be made on the device alone, and will not be reflected in the current device object. After a successful call to this method, PhysicalLocation will continue to return the 'old' value. To read the device's current name, the caller must re-load the device's description.

}

...

FIG. 37

```
...
[propget, id(DISPID_UPNPDEVICEDESCRIPTION_PRODUCTNAME),
    helpstring("a displayable string containing the product name")]
   HRESULT ProductName([out, retval] BSTR * pbstr);
        parameters:  pbstr   on return, the address of a newly-allocated string
containing the product name of the device.
        return values:        S_OK pbstr contains a newly-allocated string that must
be freed when no longer needed.

   [propget, id(DISPID_UPNPDEVICEDESCRIPTION_DESCRIPTION),
    helpstring("displayable summary of the device's function")]
   HRESULT Description([out, retval] BSTR * pbstr);
        parameters:  pbstr   on return, the address of a newly-allocated string
containing a short description of the device meaningful to the user.
        return values:        S_OK pbstr contains a newly-allocated string that must
be freed when no longer needed.

   [propget, id(DISPID_UPNPDEVICEDESCRIPTION_MODELNAME),
    helpstring("displayable model name")]
   HRESULT ModelName([out, retval] BSTR * pbstr);
        parameters: pbstr    on return, the address of a newly-allocated string
containing the manufactuer's model name of the device.
        return values: S_OK         pbstr contains a newly-allocated string that must
be freed when no longer needed.

   [propget, id(DISPID_UPNPDEVICEDESCRIPTION_SERIALNUMBER),
    helpstring("displayable serial number")]
   HRESULT SerialNumber([out, retval] BSTR * pbstr);
        parameters: pbstr    on return, the address of a newly-allocated string
containing the manufacturer's serial number of the device.
        return values: S_OK         pbstr contains a newly-allocated string that must
be freed when no longer needed.
        note: a device's serial number is not guaranteed to be globally unique.  The
DeviceUniqueName should always be used to distinguish devices.

   [propget, id(DISPID_UPNPDEVICEDESCRIPTION_MANUFACTURERNAME),
    helpstring("displayable manufacturer name")]
   HRESULT ManufacturerName([out, retval] BSTR * pbstr);
        parameters
...
```

FIG. 38

...
pbstr, on return, the address of a newly-allocated string containing the name of the device's manufactuer.
return values: S_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_MANUFACTURERURL),
helpstring("URL to the manufacturer's website")]
HRESULT ManufacturerURL([out, retval] BSTR * pbstr);
parameters: pbstr, on return, the address of a newly-allocated string containing the URL of the manufacturer's website.
return values: S_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_MODELNAME),
helpstring("displayable model name")]
HRESULT ModelName([out, retval] BSTR * pbstr);
parameters: pbstr, on return, the address of a newly-allocated string containing the manufactuer's model name for the device.
return values: S_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_SUPPORTLIST),
helpstring("technical support contact information")]
HRESULT SupportList([out, retval] BSTR * pbstr);
parameters: pbstr, on return, the address of a newly-allocated, multi-line string containing phone numbers and other information that can guide the user to technical support. This string must be freed when no longer needed.
return values: S_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_FAQLIST),
helpstring("FAQ access display information")]
HRESULT FAQList([out, retval] BSTR * pbstr);
parameters: pbstr, on return, the address of a newly-allocated, multi-line string containing FAQ information that can provide the user with URLs at which device FAQs may be located.
return values: S_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.
...

FIG. 39

```
...
[propget, id(DISPID_UPNPDEVICEDESCRIPTION_UPDATELIST),
    helpstring("information explaining where the user can update the device's
firmware")]
   HRESULT UpdateList([out, retval] BSTR * pbstr);
        parameters: pbstr, on return, the address of a newly-allocated, multi-line
string containing information and URLs from which the user can download updates
for the device's firmware.
        return values: S_OK, pbstr contains a newly-allocated string that must be
freed when no longer needed.
```

FIG. 40

```
[
object,
uuid(FDBC0C73-BDA3-4C66-AC4F-F2D96FDAD68C),
dual,
helpstring("IUPNPDevices Interface"),
pointer_default(unique)
]
IUPNPPropertyBag
{

   [propget, id(DISPID_UPNP_PROPERTYBAG_READ),
    helpstring("reads a value from the property bag")]
   HRESULT Read([in] BSTR bstrName, [out, retval] VARIANT * pvarResult);
        parameters: bstrName, name of the property to read.  case is ignored.
pvarResultvalue of the property.  if the property doese not exist, this is of type
VT_EMPTY
        return values: S_OK, the value was found in the property bag, and returned
in pvarResult.  S_FALSE, there was no value with the given name in the property
bag.  *pvarResult is of type VT_EMPTY

   [propget, id(DISPID_UPNP_PROPERTYBAG_WRITE),
    helpstring("writes a value to the property bag")]
   HRESULT Write([in] BSTR bstrName, [in] VARIANT * pvarValue);
        parameters: bstrName, name of the property to write.  case is preserved
when writing.  The supplied value will replace any other values of the same name,
even if they differ in case.  pvarValue, value of the property to write.
        return values: S_OK, the value was written to the property bag, replacing the
value currently associated with this property, if it existed.

   [propget, id(DISPID_UPNP_PROPERTYBAG_DELETE),
    helpstring("removes a value from the property bag")]
   HRESULT Delete([in] BSTR bstrName);
        parameters: bstrName, name of the value to remove from the property gab.
case is ignored when finding a value to remove.
        return values: S_OK, the value has been removed from the property bag.
S_FALSE, the value was not found in the property bag.

};
```

FIG. 41

```
[
object,
uuid(A295019C-DC65-47DD-90DC-7FE918A1AB44),
dual,
helpstring("IUPNPService Interface"),
pointer_default(unique)
]
interface IUPNPService : IDispatch
{
[id(1), helpstring("method GetProperty")]
HRESULT GetProperty(
[in] BSTR bstrPropertyName,
[out, retval] VARIANT *pValue
);

[id(2), helpstring("method InvokeAction")]
HRESULT InvokeAction(
[in] BSTR bstrActionName,
[in] VARIANT saActionArgs,
[out, retval] long *plStatus
);

[propget, id(3), helpstring("property DCPI")]
HRESULT DCPI(
[out, retval] BSTR *pVal
);

   [propget, id(4),
    helpstring("returns a manufactuer-defined extension property")]
   HRESULT VendorExtension([out, retval] VARIANT * pvarValue );
        parameters: pvarValueOn return, this variant is filled with the value of the
"extension" element.  If none exists, pvarValue is set to VT_EMPTY
        return values: S_OK, varValue is set to the extension element.  S_FALSE,
no vendor extension element exists.  pvarValue is VT_EMPTY
```

FIG. 42

```
[
object,
uuid(FDBC0C73-BDA3-4C66-AC4F-F2D96FDAD68C),
dual,
helpstring("IUPNPDevices Interface"),
pointer_default(unique)
]
   interface IUPNPDevices : IDispatch
{
[propget, id(1), helpstring("property Count")]
HRESULT Count(
[out, retval] long *pVal
);

[propget, id(DISPID_NEWENUM), helpstring("property _NewEnum")]
HRESULT _NewEnum(
[out, retval] LPUNKNOWN *pVal
);

[propget, id(DISPID_VALUE), helpstring("property Item")]
HRESULT Item(
[in] long lIndex,
[out, retval] VARIANT *pVal
);
};
```

FIG. 43

```
[
object,
uuid(3F8C8E9E-9A7A-4DC8-BC41-FF31FA374956),
dual,
helpstring("IUPNPServices Interface"),
pointer_default(unique)
]
interface IUPNPServices : IDispatch
{
[propget, id(1), helpstring("property Count")]
HRESULT Count(
[out, retval] long *pVal
);

[propget, id(DISPID_NEWENUM), helpstring("property _NewEnum")]
HRESULT _NewEnum(
[out, retval] LPUNKNOWN *pVal
);

[propget, id(DISPID_VALUE), helpstring("property Item")]
HRESULT Item(
[in] long lIndex,
[out, retval] VARIANT *pVal
);
};
```

FIG. 44

```
<contract>

 <protocol id="protocolDef">
  <HTTP version="1.1">
   <URL> http://investor.msn.com/stockquote </URL>
   <M-POST>
    <MAN> http://www.upnp.org/service-control/m-post </MAN>
   <M-POST>
   <HEADER name="Content-Type" value="text/xml" />
  </HTTP>
 </protocol>

 <RequestResponse name="getQuote">
  <protocol is="protocolDef" />
  <in      is="symbol" />
  <out     is="stockQuote" />
  <error   is="error" />
 </RequestResponse>

 <RequestResponse name="getQuotes">
  <protocol is="protocolDef" />
  <in      is="symbols" />
  <out     is="stockQuotes" />
  <error   is="error" />
 </RequestResponse>

 <!-- // schema definition follows -->

 <schema xmlns="urn:schema-microsoft-com:xml-data"
       xmlns:dt="urn:schema-microsoft-com:datatypes">

  <ElementType name="symbol" dt:type="string" />

  <ElementType name="symbols">
   <element type="symbol" maxOccurs="*" />
  </ElementType>

  <ElementType name="stockQuote">
   <element type="company" />
   <element type="ticker" />
...
```

FIG. 45

```
...
    <element type="previousClose" />
    <element type="openingTrade" />
    <element type="lastTrade" />
    <element type="volume" />
  </ElementType>

  <ElementType dt:type="string" name="company" />
  <ElementType dt:type="string" name="ticker" />
  <ElementType dt:type="string" name="previousClose" />
  <ElementType dt:type="string" name="openingTrade" />
  <ElementType dt:type="string" name="lastTrade" />
  <ElementType dt:type="string" name="volume" />

  <ElementType name="stockQuotes">
    <element name="stockQuote" maxOccurs="*" />
  </Element>

  <ElementType name="error">
    <element type="reason" />
  </ElementType>

  <ElementType dt:type="string" name="reason" />

 </schema>

</contract>
Request for "getQuote"

M-POST /stockquotes HTTP/1.1
Host: amarg5:8586
Content-Type: text/xml
Man: "http://www.upnp.org/service-control/m-post"; ns=01
01-MethodName: getQuotes
01-MessageType: Call
Accept-Language: en-gb, en;q=0.8
Referer: http://amarg5/uPnPService/Services/Stock/Client/ticker.htm
Content-Length: 327
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Connection: Keep-Alive

...
```

FIG. 46

```
...

<symbol>MSFT</symbol>
Response for "getQuote"

HTTP/1.1 200 OK
Connection: close
Cache-Control: private
Date: Mon Aug 16 15:37:35 PDT 1999
Expires: Mon Aug 16 15:37:35 PDT 1999
Content-Type: text/xml
Content-Length: 7912
Man: "http://www.upnp.org/service-control/m-post"; ns=01
Ext:
01-MessageType: CallResponse

<stockQuote>
  <company>Microsoft%20Corporation</company>
  <ticker>MSFT</ticker>
  <previousClose>84%2011/16</previousClose>
  <openingTrade>85%201/16</openingTrade>
  <lastTrade>84%205/16</lastTrade>
  <volume>28.66%20Mil</volume>
</stockQuote>
```

FIG. 47

```
<!-- XDR Schema for protocol section of contract -->

<schema name="contract"
      xmlns="urn:schema-microsoft-com:xml-data"
      xmlns:dt="urn:schema-microsoft-com:datatypes">

 <ElementType name="contract"
          xmlns:protocolNS="contract-protocol"
            xmlns:msgPatternNS="contract-msgPatterns"
            xmlns:schemaNS="urn:schema-microsoft-com:xml-data">

   <element type="protocolNS:protocol" />

   <element type="msgPatternNS:RequestResponse" minOccurs="0"
maxOccurs="*" />
   <element type="msgPatternNS:SolicitResponse" minOccurs="0" maxOccurs="*"
/>

   <element type="schemaNS:schema"          minOccurs="0" maxOccurs="*" />

 </ElementType>

</schema>
...
```

FIG. 48

```
...
Protocol
<!-- XDR Schema for protocol section of contract -->

<schema name="contract-protocol"
      xmlns="urn:schema-microsoft-com:xml-data"
      xmlns:dt="urn:schema-microsoft-com:datatypes">

 <ElementType name="protocol">

  <!-- ID -->
  <AttributeType name="id" dt:type="id" />
  <Attribute type="id" />

  <group order="one">
   <element xmlns:http="contract-protocol-HTTP" type="http:HTTP" />
   <element xmlns:gena="contract-protocol-GENA" type="gena:GENA" />
   // other protocol definitions go here
  </group>

 </ElementType>

</schema>
...
```

FIG. 49

```
...
HTTP
<!-- XDR Schema for HTTP section of contract -->

<schema name="contract-protocol-HTTP"
      xmlns="urn:schema-microsoft-com:xml-data"
      xmlns:dt="urn:schema-microsoft-com:datatypes">

 <ElementType name="HTTP">

  <!-- HTTP version -->
  <AttributeType name="VERSION" dt:type="string" default="1.1" />
  <Attribute type="VERSION" />

  <!-- The Verb to use -->
  <group order="one">
   <element type="GET"    />
   <element type="POST"   />
   <element type="M-POST" />
  </group>

  <!-- The protocol data -->
  <element type="URL"    />
  <element type="QUERY"  minOccurs="0" />
  <element type="HEADER" minOccurs="0" />

 </ElementType>

 <ElementType name="URL" dt:type="string" />

 <ElementType name="QUERY">
  <attribute type="name"     />
  <attribute type="value"    />
  <attribute type="required" />
 </ElementType>

...
```

FIG.50

```
...

<ElementType name="HEADER">
  <attribute type="name"  />
  <attribute type="value" required="yes" />
</ElementType>

<!-- Verb declarations -->
<ElementType name="GET"/>

<ElementType name="POST">
  <element type="PARAM" minOccurs="0" maxOccurs="*" />
</ElementType>

<ElementType name="PARAM">
  <element type="name" />
  <element type="default"  />
  <element type="value"    />
  <element type="required" />
</ElementType>

<AttributeType name="name"     dt:type="string"  required="yes" />
<AttributeType name="default"  dt:type="string"  />
<AttributeType name="value"    dt:type="string"  />
<AttributeType name="required" dt:type="boolean" default="no" />

<ElementType name="M-POST">
  <element type="MAN" />
</ElementType>

<ElementType name="MAN" dt:type="string" />

</schema>
```